The Dissertation Committee for Aymeric-Pierre Bernard Peyret
certifies that this is the approved version of the following dissertation:

Morphodynamics and geometry of channels, turbidites, and bedforms

Committee:

David Mohrig, Supervisor

Gary Kocurek

Wonsuck Kim

Larry W. Lake

Craig Fulthorpe

Morphodynamics and geometry of channels, turbidites, and bedforms

by

Aymeric-Pierre Bernard Peyret, Dipl. Ing., M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2011

To my family and friends

Acknowledgments

The author wishes to express his gratitude to Dr. David Mohrig for his valuable support and accurate supervision in the preparation of this dissertation and throughout the past few years at U.T., and to acknowledge Drs. Gary Kocurek, Wonsuck Kim, Karry W. Lake and Craig Fulthorpe for their presence in the dissertation committee.

I would like to thank the faculty and staff of the Department of Geological Sciences, Jackson School of Geosciences, for their guidance, as well as Drs. Mohrig and Kocurek's present and past research groups for their support and advice.

This project was also made possible by my family and friends, who provided me with their moral or financial support .

AYMERIC-PIERRE, BERNARD PEYRET

The University of Texas at Austin

December 2011

Morphodynamics and geometry of channels, turbidites, and bedforms

Publication No. _____

Aymeric-Pierre Bernard Peyret, Ph.D.
The University of Texas at Austin, 2011

Supervisor: David Mohrig

The evolution of landscapes and seascapes in time is the result of the constant interaction between flows and topography. Flows change topography, which in turn change the flow. This feedback causes evolution processes to be highly non-linear and complex. When full analytical derivations of the co-evolution of topography and flow are not possible without oversimplifications, as is the case in river bends, recent large topographical datasets and modern computers allow for correlations between horizontal (planview) and cross-sectional geometry of channels. Numerical analysis in the Mississippi and Trinity rivers indicate that the type of correlation between river radius of curvature and bankfull channel width depends on the migration behavior of the river. In other cases, channel topography may only have a second-order effect on its own evolution, as is the case for fully depositional turbidity currents, and the evolution of æolian field topography may only be a function of this topography. I show that in these situations, changes in topography may be decoupled from details of the flow field and modeled very easily with a good accuracy.

Table of Contents

List of Tables

List of Figures

Introduction

Morphodynamics is the study of the evolution of landscapes and seascapes through the constant interaction of topography and sediment-laden flows[1]. Channels act as flow guides and modify flows, whereas flows deposit and/or entrain sediment from the channel, thereby modifying its shape. Traditional morphodynamics relies heavily on fluid dynamics to model the flows, and on a set of (for the most part) empirical equations to connect entrainment and deposition to both the flow and the bed. This type of modeling results in large systems of differential equations that may not be solved without simplifying assumptions on the shape of the channel or on the properties of the flow, and analytical solutions cannot be found without further simplifications and assumptions. The $\{\text{topography}, \text{flow}\}$ system is a closed-loop system:

- I assume that the influence of life on channel- and bedform-shapes is minimal, except for human life in some areas; While rates of surface evolution are affected by life, there is no unique geomorphic signal of life [Dietrich and Perron, 2006]
- When human intervention results in modifications of riverbeds or dunes, the original $\{\text{topography}, \text{flow}\}$ system is destroyed and another is created, with different initial conditions

[1] Definition adapted from Parker [2006]

- External influences which act as external perturbations on the system include weather and climate, as well as long- and short-term geological processes, such as subsidence, sea-level changes and earthquakes. In the majority of cases, river evolution is between the shortest (eathquakes, rapid soil compaction) and the longest (sea-level changes) time scales of geological processes. Events with a shorter time scale, in a way similar to human influence, will break the time continuum of the $\{\text{topography}, \text{flow}\}$ system and restart a system with new initial conditions, while events with longer time scales will have only a second-degree influence on a $\{\text{topography}, \text{flow}\}$, similar to amplitude modulation radio systems where the carrier frequency, a low frequency wave, does not significantly change the signal frequency

- Other sources of external forcings, such as gravity and Coriolis acceleration, may be supposed constant for the space- and time-scales of our study

- Therefore, we may assume that only the flow modifies the topography, and only the topography modifies the flow

We may therefore want to assume that the $\{\text{topography}, \text{flow}\}$ closed loop system may be uncoupled. Just like for a small (R, L, C) electrical circuit, or the response of an air-conditioning system to a change in room temperature, we may assume that after a short transient response triggered by a change in conditions (or forcings), a steady state is established until the next change in external conditions, and that at this point it is possible to study both elements of the system independently. The definition of steady state in this case depends on the time scale that is considered: a

channel that appears immobile with the naked eye may in fact be migrating rapidly over the course of a few years.

Time-scale effects in closed-loop systems are not the only reasons why flow, topography and topograpic evolution may be uncoupled. Intrinsic properties of the flow, as is the case the turbidity currents and dunes, may at least partially allow one to treat the flow independently of the topography, or the evolution of topography independently of topography itself. In the turbidity current case, as I will show in the first chapter of this dissertation, the length scales over which sand grains are transported in the flow without touching ground may be extremely large compared to the local topography and therefore deposition patterns are largely independent of bed topography. In the æolian dune field case, these length scales are so small that the topography evolves primarily as a function of itself and by local fluctuations of the flow. The second chapter of this dissertation makes use of this property to quickly simulate the evolution of a cross-section of a dune field and study statistical properties of the field as a whole.

The third and final chapter presents a finite-difference code[2] that computes the radius of curvature of a channel centerline in planview and is used to correlate this radius with cross-sectional geometrical parameters, such as the bottom slope or the channel width. I assume in this chapter that the $\{\text{channel}, \text{flow}\}$ system is in a steady-state over a period of time long enough to neglect the rapid variations of

[2] A dissertation is not a practical publishing means for computer code. Therefore, no code listing is presented in this document, but computer files that are necessary to run the various codes presented in this dissertation are available directly from the author or at `http://doi.pangaea.de/10.1594/PANGAEA.771541`.

the flow; I also assume that either this period of time is short enough to neglect the migration, or that migration does not affect these correlations.

Chapter 1

Geometry of deposition in turbidity currents

Introduction and motivation

"The essence of morphodynamics is in the interaction between the flow and the bed. The flow changes the bed, which in turn changes the flow." The phase shift between the effects of the topography on the flow and the effects of the flow on the topography depends on the environment, and particularly on the distance that grains may travel without touching ground. In Æolian processes, the relatively low density and viscosity of air imply that the distance sand grains are advected in the air is much shorter than large-scale topographic features (dunes). In this case, the sediment flux, and therefore its space derivative (erosion or deposition), is a function of the geometry of the topography (bed elevation and slope), and the details of the flow turbulence only have a second-order effect on topography [Jerolmack and Mohrig, 2005]. Conversely, we will show in this chapter that the length that suspended grains are advected in depositional turbidity currents may be much longer than local topography, effectively uncoupling the evolutions of the flow and the topography, thereby allowing for simplified deposition models.

This chapter describes tentative means to simplify models of fully depositional turbidity currents so we can estimate deposition very easily. The theory for

the simplifications is based on the concept of advection length, which we define as the average distance that a grain of a given size will spend in a flow of constant velocity, density and viscosity before it it deposited. We will extend this notion by introducing a "composite advection length" which allows for slight changes in velocity and density of the turbidity current. This depositional turbidity current model explores the domain of very large advection length, where the evolution of the flow is quasi-independent of the topography while the Æolian dune field model that we present in chapter 2 explores the other end of the advection length range, where the evolution of topography is quasi-independent of the evolution of the flow, by using simplified shapes to assess the effect of the interactions between bedforms.

Why do we need to simplify models? Current modeling tools compute the whole flow field, which creates complex code (to create, debug, modify) that in turn results in computationally intensive programs. These models described in this proposal are solved efficiently for large time and space scales. We will first describe literature cases that motivate our work on this topic, then derive a model and show the way it is implemented in practice, then finish with a comparison between model and measurements.

1.1 Study of depositional turbidity currents filling sinuous submarine channels

The depositional turbidity current model is mainly motivated by three literature cases (García [1994], Lamb et al. [2010] and Straub et al. [2011b]), which show that a complicated flow field does not always transcribe to a very complicated

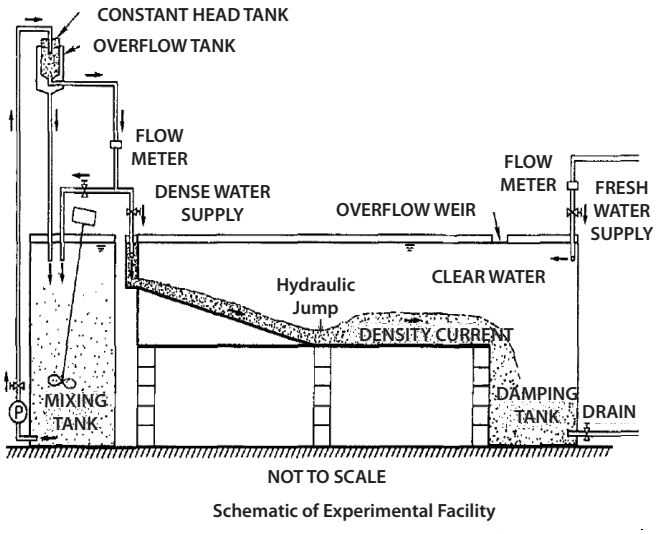


Figure 1.1: Schematic of García's [1993, 1994] experimental facility

deposition pattern. Therefore, we ask the question: what if, for certain cases, we could avoid computing the whole flow field?

Let us first show some examples that motivate this question. We begin with Marcelo García's experiments, as described in his 1994 paper. As shown on Figure 1.1, the turbidity current produced by García's experiments went through a hydraulic jump about the bottom slope break. Figure 1.2 shows complex velocity profiles recorded from García's experiments and commonly found in turbidity currents. However, as we can see on Figures 1.3 and 1.4, the depositional patterns and the variation of median grain size with distance from inlet are very smooth profiles and do not show any of the complexity of the flow, thus demonstrating the disconnection between flow field and deposit.

Another example of this behavior may be observed in Lamb et al.'s [2010]

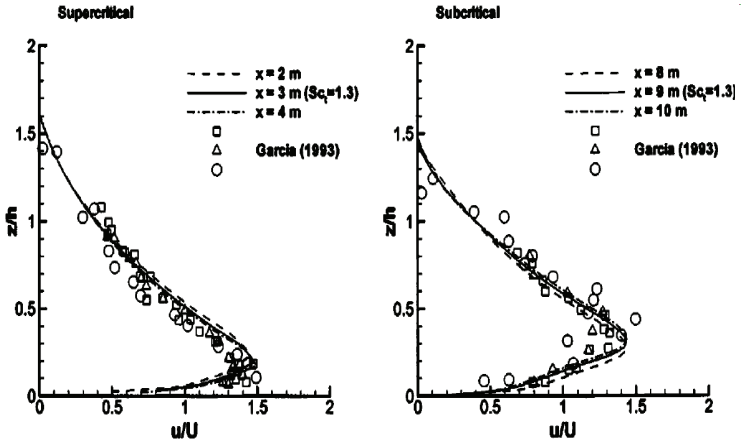


Figure 1.2: Velocity profiles for García's [1993] experiments. The various lines represent these velocity profiles at different locations downstream.

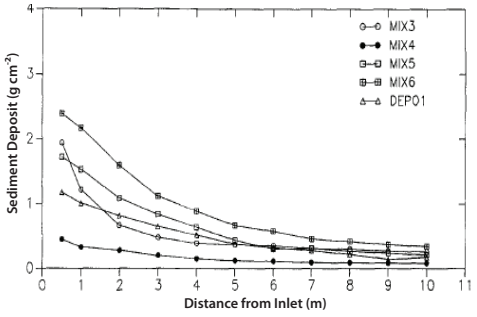


Figure 1.3: Depositional patterns produced by currents driven by poorly sorted sediment from García's [1994] experiments

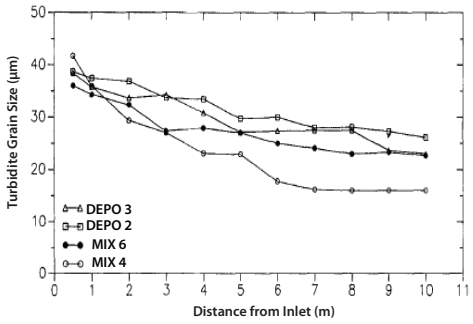


Figure 1.4: Variation of sediment deposit median grain size D_{50} with distance from inlet, from García's [1994] experiments

experiments. Figure 1.5 shows the plunging of a decelerating current, followed by an accelerating turbidity current. Velocities in these currents are given at the top of Figure 1.6, and display trends (deceleration, then acceleration) that are neither found in the deposition rates (profiles circled in red, in the middle plot of Figure 1.6) nor in the median grain size profiles (bottom plots of Figure 1.6). Once again,

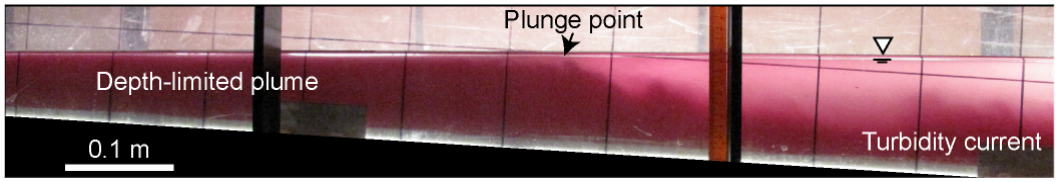


Figure 1.5: Lamb's picture of plunging density current [Lamb et al., 2010]

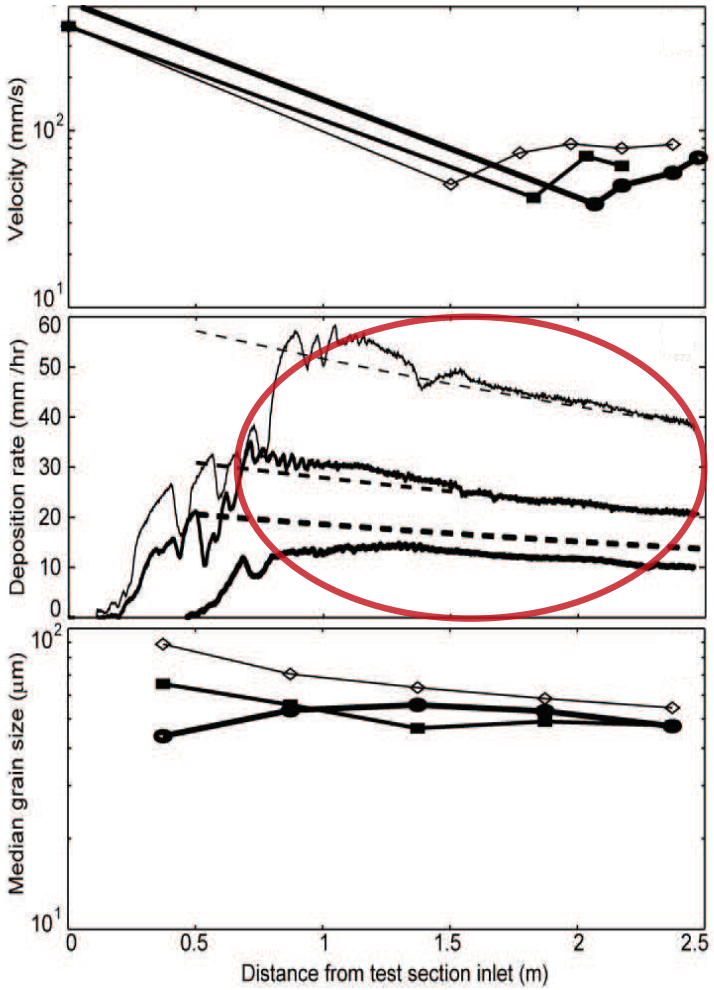
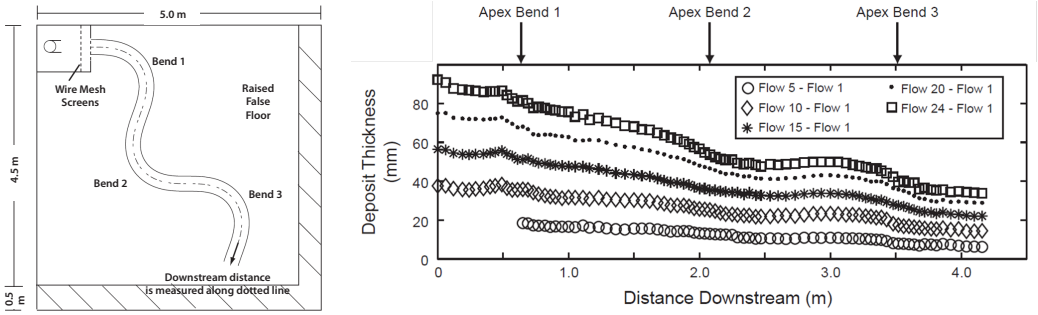


Figure 1.6: Velocity, deposition rate and median grain size as functions of distance from test section inlet in Lamb's experiments [Lamb et al., 2010]

the trends in the flow field, and the trends in the resulting deposits and grain sizes are not correlated.



(a) Experimental setup used in [Straub et al., 2011b]

(b) Depositional patterns recorded by [Straub et al., 2011b]

Figure 1.7: The experiments presented in [Straub et al., 2011b] show once again very simple deposition patterns with a complex flow and channel geometry

Our last example regards the Straub et al. [2011b] turbidity currents running in a sinuous channel as shown in Figure 1.7(a). Sinuous channels, whether subaerial or subaqueous, are characterized by very complex flow in the bends. However, the deposits shown in Figure 1.7(b) do not show – again – any of the complications of the channel. How can simple physics and geometry explain this lack of correlation between flow, channel (flow guide) and deposits? We will now derive and describe a simple method to compute deposition profiles.

1.1.1 Physical model and computational transcription

Let us begin with the physics. We can relate deposition rate and sediment flux in 1D using equation (1.1.1):

$$\frac{\partial \eta}{\partial t} = -\frac{1}{1-\lambda_p} \cdot \frac{\partial q_s}{\partial x} \tag{1.1.1}$$

where λ_p is the bed porosity, η the bed elevation and q_s the volumetric sediment flux, t time and x the distance in the direction of the water flux. For a single sediment size, the change in bed elevation may also be expressed as a function of particle fall velocity w_s, near-bed sediment concentration c_b, and a dimensionless entrainment parameter e, as shown in equation (1.1.2).

$$\frac{\partial \eta}{\partial t} = \frac{w_s}{1-\lambda_p} \cdot (c_b - e) \tag{1.1.2}$$

Consequently, the sediment flux may be written as a function of the same parameters as in equation (1.1.3):

$$\frac{\partial q_s}{\partial x} = e \cdot w_s - c_b \cdot w_s \tag{1.1.3}$$

As we are studying fully depositional currents, the entrainment e is zero, and therefore

$$\frac{\partial q_s}{\partial x} = -w_s \cdot c_b \tag{1.1.4}$$

Given that the water flow discharge and concentration are more natural parameters than sediment fluxes, we introduce a few other variables: c the depth-averaged sediment concentration, q the flow discharge per unit width, c_c the depth-averaged sediment concentration at capacity[1] and q_{sc} the sediment transport capacity per unit width. These new variables are related through equations (1.1.5) and

[1] We define capacity as the flow conditions such that the flux of sediment being transported by the flow at some location is equal to the sediment flux that would be predicted by the local measure of boundary shear stress and particle size, fluid and sediment density, etc. at that location.

(1.1.6).

$$q_s = c \cdot q \tag{1.1.5}$$

$$q_{sc} = c_c \cdot q \tag{1.1.6}$$

Equation (1.1.4) may therefore be rewritten

$$\frac{\partial (c \cdot q)}{\partial x} = -w_s \cdot c_b \tag{1.1.7}$$

If we suppose that the flow discharge in constant in space, we obtain

$$q \cdot \left(\frac{\partial c}{\partial x} \right) = -w_s \cdot c_b \Longrightarrow \frac{\partial c}{\partial x} = -\frac{w_s \cdot c_b}{q} \tag{1.1.8}$$

We introduce another variable r_0 that relates the near-bed sediment concentration c_b to the depth-averaged sedimnent concentration c with $r_0 = \frac{c_b}{c}$. r_0 is often close to 2 (see Lamb et al. [2010]); The previous equation becomes

$$\frac{\frac{\partial c}{\partial x}}{c} = -\frac{r_0 \cdot w_s}{q} \tag{1.1.9}$$

or

$$\frac{\partial \ln (c(x))}{\partial x} = -\frac{r_0 \cdot w_s}{q} \tag{1.1.10}$$

We integrate this equation over the distance from x_a (initial point) to $x \geq x_a$ where we solve for the suspended sediment concentration

$$\ln \left[\frac{c(x)}{c(x_a)} \right] = -\frac{r_0 \cdot w_s}{q} (x - x_a) \tag{1.1.11}$$

i.e.

$$\frac{c(x)}{c(x_a)} = \exp \left[-\frac{r_0 \cdot w_s}{q} (x - x_a) \right] \tag{1.1.12}$$

If we define the advection length for a single grain size (or "simple advection length") as $L_a = \left(\frac{q}{r_0 \cdot w_s}\right)$, then we have

$$c(x) = c(x_a) \cdot \exp\left(-\frac{x - x_a}{L_a}\right) \tag{1.1.13}$$

and

$$q_s(x) = q \cdot c(x) = q \cdot c(x_a) \cdot \exp\left(-\frac{x - x_a}{L_a}\right) \tag{1.1.14}$$

At capacity, the near-bed sediment concentration (bed property) counterbalances the entrainment (flow property), so that:

$$\left.\begin{aligned} \frac{\mathrm{d}q_s}{\mathrm{d}x} = w_s \cdot (e - c_b) = 0 \Longrightarrow e = c_b \\ c = c_c \end{aligned}\right\} \Longrightarrow c_c = \frac{e}{c_b} \cdot c = e \cdot \underbrace{\frac{c}{c_b}}_{\frac{1}{r_0}} \Longrightarrow \underbrace{q \cdot c_c}_{q_{sc}} = q \cdot \frac{e}{r_0} \tag{1.1.15}$$

Therefore, if we combine equations (1.1.1), (1.1.3) to (1.1.15), we obtain

$$\frac{\mathrm{d}q_s}{\mathrm{d}x} = w_s \cdot (e - c_b) = w_s \cdot \left(\frac{r_0 \cdot q_{sc}}{q} - \frac{c_b}{c} \cdot \frac{c \cdot q}{q}\right) = w_s \cdot \left(\frac{r_0 \cdot q_{sc}}{q} - \frac{r_0 \cdot q_s}{q}\right) \tag{1.1.16}$$

Reorganizing equation (1.1.16) yields

$$\left(\frac{q}{r_0 \cdot w_s}\right) \frac{\mathrm{d}q_s}{\mathrm{d}x} = q_{sc} - q_s \tag{1.1.17}$$

$$\boxed{L_a \frac{\mathrm{d}q_s}{\mathrm{d}x} = q_{sc} - q_s} \tag{1.1.18}$$

Using equation (1.1.18), we look at two end members by comparing L_a to our length scale of interest, $x - x_a$:

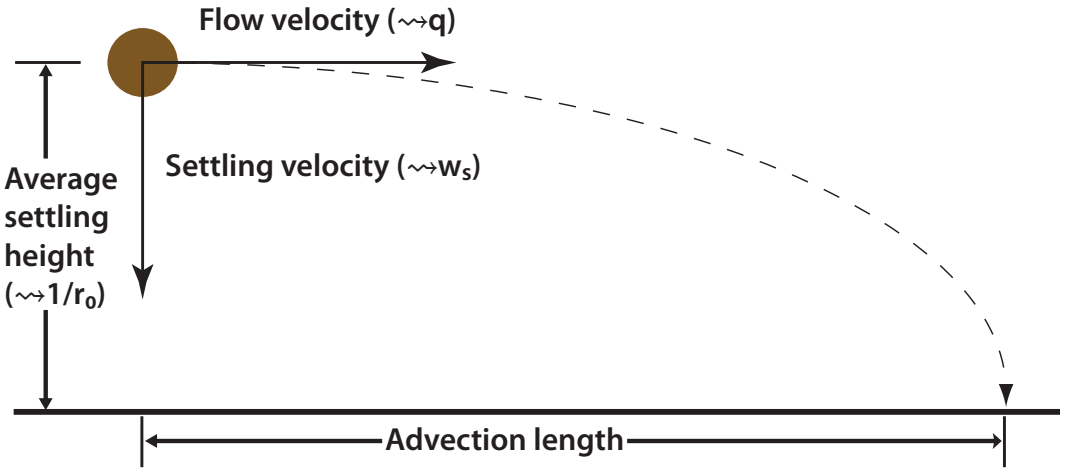


Figure 1.8: Schematic representation of the advection length for a single grain size [Lamb et al., 2010]

- when the advection length is smaller than the length scale of interest ($L_a \to 0$), $q_s(x) \to q_{sc}(x)$ and $\frac{d\eta}{dt} = -\frac{1}{1-\lambda_p} \cdot \frac{dq_{sc}}{dx}$, which means that deposition follows the local transport capacity gradient, and therefore the local flow dynamics and topography. Modeling this case requires a more complete description of the flow field;

- when the advection length is larger than the length scale of interest ($L_a \to +\infty$), equation (1.1.14) shows that $q_s(x) = q_s(x_a)$, which means that deposition follows the inlet sediment flux, and modeling deposition requires only the simplest description of the flow field.

We can think about the advection length as the "sediment inertia", i.e. the balance between flow velocity and particle settling velocity, given a certain initial height, as represented in Figure 1.8. The advection length is a statistical variable, because we assume that grains settle from the center of suspended-sediment mass

of the current; grains located below that center of mass will travel smaller distances without touching ground, grains located above that center of mass will travel larger distances without touching ground, but on average grains will travel a distance close to L_a without touching ground. The average settling height is proportional to $\frac{1}{r_0}$ (see appendix, section A.4.1 for an equation of the center of mass of suspended sediment in a Rouse profile), the flow velocity is proportional to q and the settling velocity is w_s, which are the three variables that are used to define the advection length. This advection length, which we call "simple advection length" is subject to limitations in its use, as it is valid for a grain in a constant velocity, constant density, constant viscosity flow, which means that larger grain sizes are not present in the flow (their deposition would change the density, viscosity and velocity of the fluid). Simple advection lengths are computed as shown in Figure 1.9. Input variables (in yellow and orange, inside parallelograms) are $\overline{u}$ (depth-averaged flow velocity), D_{50} (median grain size) and h (flow depth). Constants are in circles: g (gravitational acceleration), κ (von Kármán's constant), C_f (drag coefficient), ν (water dynamic viscosity, in case the water temperature is not given). In this figure, $Z_\star$ is the center of mass for a Rouse concentration profile for a single particle size. L_a (in the green oval) is the resulting simple advection length.

To avoid the limitations listed above, we extend the notion of advection length to a grain within a distribution of multiple grain sizes, in a flow where the deposition of larger grains changes velocity and density (viscosity changes are negligible). However, we first need to make a few assumptions. We assume that flows are fully depositional, which corresponds to many primary hydrocarbon reservoir

objectives (such as channel channel-filling deposits and levees), and that grains are perfectly spherical (Corey Shape Factor $CSF = 1$, Powers Index $P = 6$). Fall velocities are computed using Dietrich's [1982] formulæ. The concentration profiles for each grain size are assumed to follow a Rouse profile at any time (definition in appendix p.132), with the height of the bedload layer $Z_a = 10 \cdot D_{50}$, where D_{50} is the median grain size. The assumption of a Rouse profile implies a given position for the center of mass $Z_\star$, that we compute using the full distribution of grain sizes as input for Z_a but only a single grain size as input for the Rouse number p. We use the average of these centers of mass to determine a center of mass for all grains remaining in the flow, and we call it the "virtual initial height". Using the Rouse profile assumption in a fully depositional case also means that concentration decreases exponentially with distance from the inlet, because equation (1.1.14) is valid for each grain size. A value $C_f = 4 \times 10^{-3}$ is taken for the drag coefficient [Normark, 1989], and water constants (viscosity, density) are computed from temperature data using linear interpolation from tables (when temperature data are not available, standard conditions are assumed).

We call the diameters of the grains in the grain size distribution $d_1 \geq d_2 \geq \ldots \geq d_n$ (corresponding respectively to grains 1 to n). The equations given below correspond to $d_1 > d_2 > \ldots > d_n$ because given a general case with possibilities of equal grain sizes would render the equations much too complex. In the time interval $[t_{i-1}; t_i]$ between the deposition of grain $i-1$ and grain i. we suppose that all grains remaining in the flow have a simple advection length defined by the initial conditions at t_{i-1}. The trajectory of each grain size is therefore determined by its

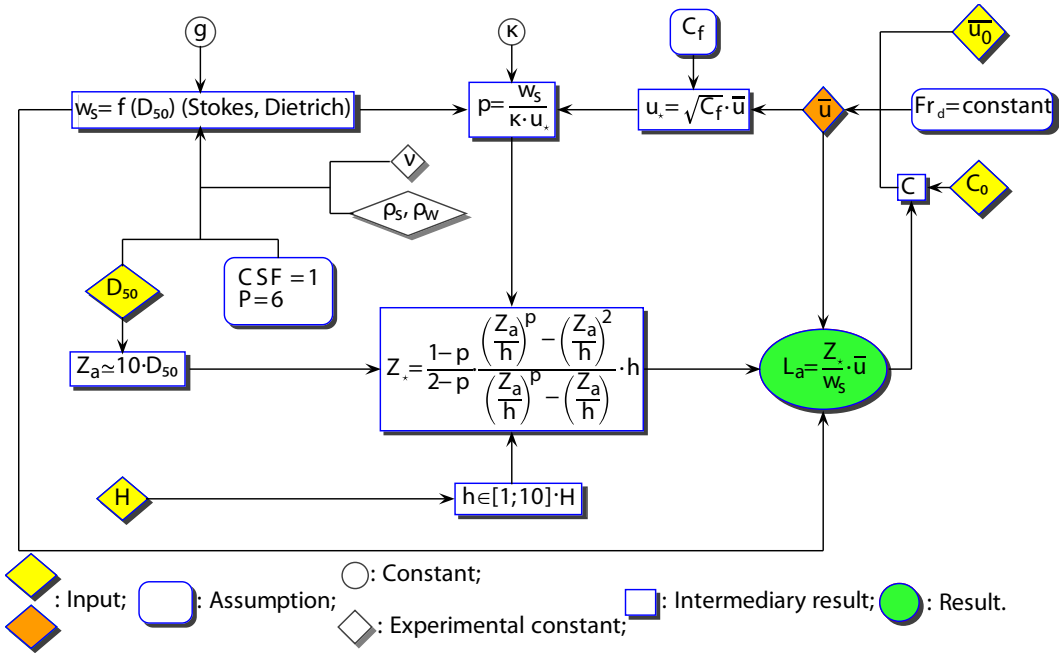


Figure 1.9: Programming "flowchart" for simple advection length. Yellow diamonds are input variables; grey circles are general constants; white diamonds are either assumed constants or other input variables (when model is used for comparisons to published literature); orange diamond is either another input variable (assumption of fixed current velocity and concentration) or an intermediary variable. rounded-corner rectangles are model assumptions; rectangles are intermediary variables; and the green oval is the output.

simple advection length during each time interval bounded by the deposition of two consecutive grain sizes, and therefore also by the deposition times of all larger grains. Equation (1.1.19) computes the composite advection length that results from these assumptions (derivation is provided in the Appendix, section A.1).

$$L_\star(n) = \frac{L_n(d_n)}{z_{n_i}} \cdot \left[z_n(t_0) - z_n(t_n) + \sum_{j=1}^{n-1} \left(\frac{z_{j+1_i}}{L_{j+1}(d_n)} - \frac{z_{j_i}}{L_j(d_n)} \right) \cdot L_\star(d_j) \right] \tag{1.1.19}$$

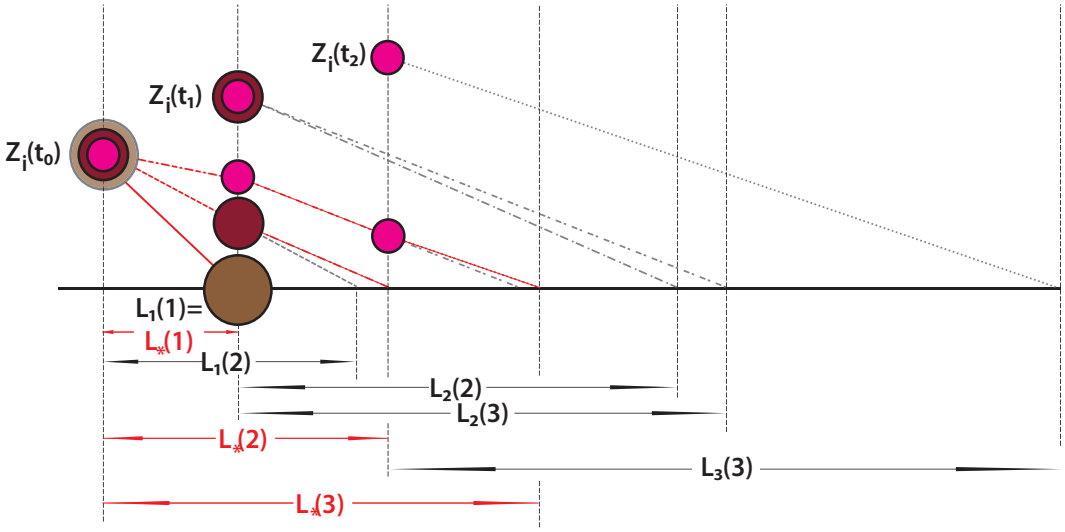


Figure 1.10: Concept of composite advection length for $d_1 > d_2 > \ldots > d_n$. Lines with the same pattern style are parallel to each other. Red lines indicate the trajectory followed by each grain. $Z_i\left(t_j\right)$ are the "virtual initial heights" of all particles remaining in the flow at time t_j. $L_j\left(i\right)$ is the simple advection length of of grain of size d_i given the initial vertical distribution of sediments at time t_{j-1}. $L_\star\left(i\right)$ is the composite advection length of a grain of size d_i.

The composite advection length $L_\star\left(d_j\right)$ for grain j therefore depends on the simple advection lengths of grain j during the various time intervals $[t_0;t_1]$, $[t_1;t_2],\ldots,\ \left[t_{j-1};t_j\right]$, where t_0 is the initial time and t_i is the time at which grain i is deposited, as pictured in Figure 1.10. Two sets of equivalent formulæ may be used to compute the composite advection length for grain j, as given by equation (1.1.20) (derivation as given in appendix p.123), when assuming that all grains have different grain sizes ($d_1 > d_2 > \ldots > d_n$; the program may be easily adjusted for a distribution that includes grains of the same size, and we have implemented that

change in our code, but formulæ cannot be written easily in a general case):

$$L_\star(n) = \frac{L_n(d_n)}{z_{n_i}} \cdot \left[z_n(t_0) - z_n(t_n) + \sum_{j=1}^{n-1} \left(\frac{z_{j+1_i}}{L_{j+1}(d_n)} - \frac{z_{j_i}}{L_j(d_n)} \right) \cdot L_\star(d_j) \right] \tag{1.1.20}$$

where

- $L_\star(n)$ is the composite advection length of grain n of diameter d_n
- $L_j(d_i)$ is the simple advection length of grain i of diameter d_i between the deposition of grain $j-1$ and j. Only the grains remaining in the flow at time t_{j-1} are taken into account for computing the center of mass, velocity, and density of the flow.
- z_{j_i} is the "Virtual initial height"of all remaining grains at time t_j; $z_{j_i} = \frac{\sum_{k=j}^{n} V_k \cdot Z_\star(k)}{\sum_{k=j}^{n} V_k}$ where V_k is the volume of grain k.
- $z_n(t_0) = z_{0_i}$
- $z_n(t_n)$ is the deposition height of grain n, in case the surface has some small irregularities.

This equation is solved recursively at each step so that the point of abscissa $L_\star(n)$ and ordinate $z_n(t_n)$ belongs to the current topography. Section A.2 of the appendix details how changes in velocity and concentration (density) of the flow are computed.

1.1.2 Results

Let us first look at the length scales for simple advection lengths (single grain size). Two plots are provided below in Figures 1.11 and 1.12.

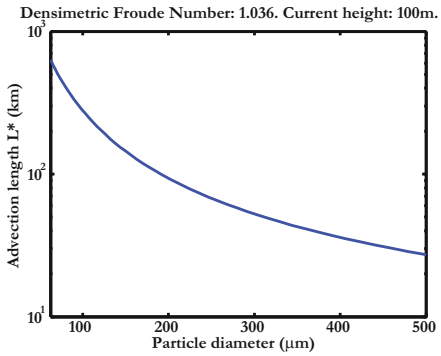


Figure 1.11: Simple advection length as a function of grain size for a constant densimetric Froude number and current height

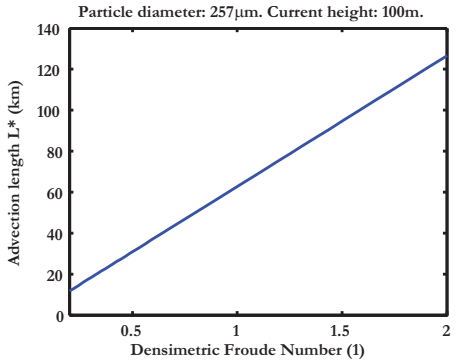


Figure 1.12: Simple advection length as a function of densimetric Froude number for a constant grain size and current height

In Figure 1.11, we assume a flow with a constant densimetric Froude number (a value commonly observed in turbidity currents), and a current height of $h = 100m$. The advection lengths for particles with diameters from $62.5\mu m$ (mud/sand limit) up to $500\mu m$ (limit coarse/very coarse sand) are in the range of tenths to hundreds of kilometers. This means that, in the flow conditions described above ($Fr_d = 1$, $h = 100m$), even coarse sand would be advected over length scales larger than many natural topographic elements. In Figure 1.12, advection lengths are plotted against densimetric Froude number for a grain size of diameter $\simeq 250\mu m$ (limit fine/medium sand) and the same current height of $100m$. The densimetric Froude number, $Fr_d = \frac{u}{\sqrt{g \cdot h \cdot \frac{\Delta\rho}{\rho}}}$ (with $\Delta\rho$ the difference between current and ambient fluid densities and ρ the density of the ambient fluid) is used here as a proxy for the flow velocity, the flow height h and the fluid densities being fixed. Figure 1.12 therefore shows that the variation of the advection length with flow velocity is linear, but even

within a large range of Froude numbers, advection lengths are enormous (between $10km$ and $130km$).

To know if we can simplify modeling, we have to compare these lengths to those of natural topographic elements. Table 1.1 has been compiled from data in the published literature and shows length scales of channels, channel levees, channel bend wavelengths. It is important to note that advection lengths follow the general direction of the flow, so only the length scales of channels in this table are important if we assume that the flow follows the dominant topography (turbidity current running down a submarine canyon), and only the widths are important if we are studying flows going, for instance, over channel levees. We can see that many of these length scales are smaller than the advection lengths computed above, which means that simplification of models is possible in many cases.

One question that remains after looking at these scales of simple advection is how much the initial distribution of grain sizes will affect the composite advection lengths. We create 18 different distributions, with 6 different shapes: 3 uniform, 3 normal, 3 lognormal, 3 gamma with scale 1 and shape 0.5, 3 gamma with scale 1 and shape 5, and 3 "reverse-lognormal" (lognormal distributions when small sizes and large sizes are switched) distributions; and with 3 different ranges of grain diameters: 6 distributions from $1\mu m$ to $100\mu m$ (no large grains), 6 distributions from $20\mu m$ to $2000\mu m$ (no very small grains) and 6 distributions from $1\mu m$ to $2000\mu m$ (all grain sizes). The cumulative distribution functions of these distributions are plotted in Figure 1.13.

Location	Topographic element		size	Reference
Monterey	Channel	Length	$300+km$	Clark and Pickering [1996b]
Monterey	Channel	Width (min)	$0.4km$	
		Depth (min)	$32m$	
Monterey	Channel	Width (max)	$2.8km$	
Monterey	Channel	Depth (max)	$884m$	
Cascadia	Channel	Length	$2000+km$	Clark and Pickering [1996b]
Cascadia	Channel	Width (max)	$5.6km$	
Cascadia	Channel	Depth (max)	$285m$	
Mississippi	Levees	Width	$600m$	Clark and Pickering [1996a]
		Depth	$10m$	
NAMOC[2]	Channel-levee system	Width	$10 < \mathbf{30} < 40km$	Skene et al. [2002]
		Depth	$100 < \mathbf{200} < 300m$	
Laurentian		Width	$16 < \mathbf{20} < 24km$	
		Depth	$300 < \mathbf{400} < 600m$	
Amazon	Channel	Wavelength	$2 < \mathbf{4.9} < 11km$	Pirmez and Imran [2003]
		Radius of curvature	$0 < \mathbf{1.05} < 3.5km$	

Table 1.1: Scales of some natural topographic elements. In red are channel longitudinal elements which one may want to compare to advection lengths of turbidity currents. In blue are channel radial (transversal) elements which may be compared to advection lengths of particles leaving the channel when going above channel levees. Other elements (in black) are provided for the reader to imagine the full scale of the channels.

[b] NAMOC stands for Northwest Atlantic Mid-Ocean Channel and is a large system of submarine channels beginning at the Hudson Strait, and where many turbidity currents run down to the abyssal plain

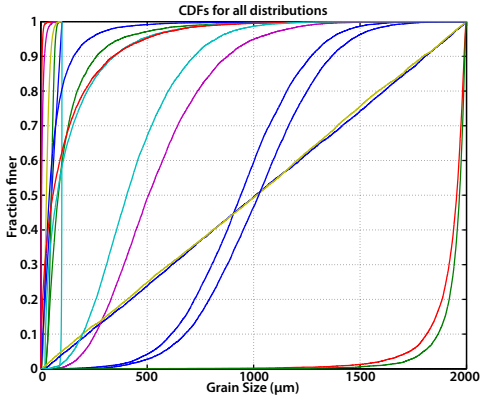


Figure 1.13: Distributions of grain sizes used to compute composite advection lengths

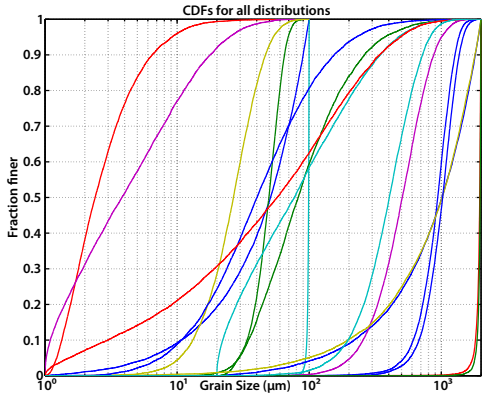


Figure 1.14: Distributions of grain sizes used to compute composite advection lengths (log scale); Colors correspond to same distributions as linear-scale plot on left

Using equation (1.1.20) and the input parameters from García's [1994] experiments (available in appendix p.132), adjusted through dimensional analysis for a slope of 3 degrees, we compute the composite advection lengths for each grain size in each distribution. The resulting grain size vs. advection length plots are given in Figure 1.15 for all input distributions. The plot shows that whatever the shape of the distribution, grain sizes and advection lengths are related through a power law for all grain sizes smaller than approximately $200\mu m$: the difference between the maximum and minimum advection lengths $\frac{\Delta A}{\overline{A}}$, over the geometric mean of these minimum and maximum $\overline{A}$, is less than 0.89 for grain sizes smaller than $324\mu m$ as shown in Figure 1.16 (1.94 when the harmonic mean is used instead of the geometric mean).

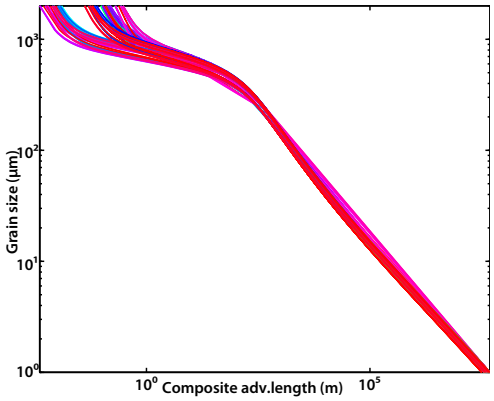


Figure 1.15: Composite advection lengths computed using García's input parameters and grain sizes from 18 distributions

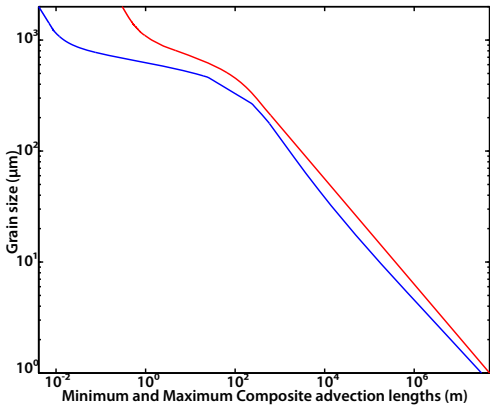


Figure 1.16: Envelope of plots given in Figure 1.15

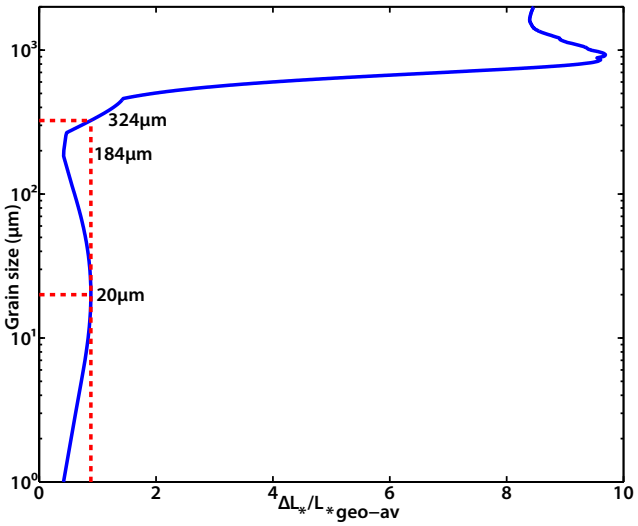


Figure 1.17: Difference $\Delta L_\star$ between maximum and minimum of envelope given in Figure 1.16, divided by the geometrical average $L_{\star_{geo-av}}$ of this minimum and this maximum

We can conclude that the initial distribution of grain sizes bears little influence on the composite advection length for grain sizes smaller than $200\mu m$. We can understand why if we look at this experiment in terms of a system with random input but a deterministic process. Smaller grains that travel over a longer distance are affected by the deterministic process more than larger grains: the smaller the grain size, the lesser the influence of the initial distribution of grain sizes.

We are now able to compare composite advection length results with published literature. We create a last distribution for which we compute composite advection lengths using García's [1994] input parameters (without slope scaling this time). The cumulative distribution function for this distribution is chose to be as close as possible to the distribution of grain sizes in García's experiments, as a comparison between Figures 1.18 and 1.19 shows. As the experimental setup cartoon showed at the beginning of this proposal (Figure 1.1), the flow goes through a hydraulic jump (which García also showed on another drawing presented here in Figure 1.20) but the median grain size of the deposit as a function of distance from inlet does not show this jump (Figure 1.4. Comparing this median grain size deposit plot to the plot of grain size vs. composite advection lengths that we obtain using the same input data (Figure 1.21) demonstrates the accuracy of using the advection length to simplify modeling in this case.

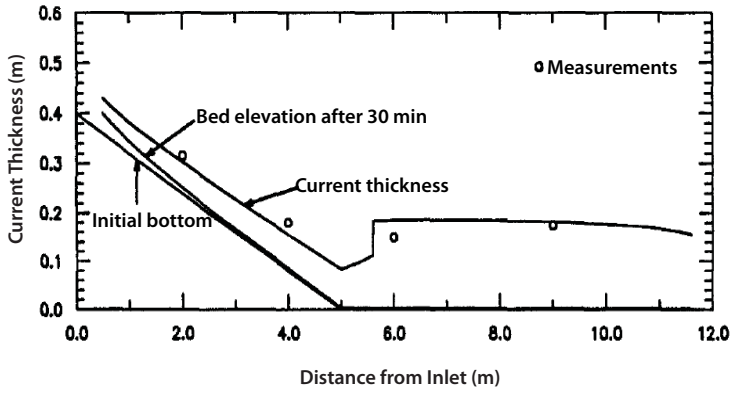


Figure 1.20: García's [1994] cartoon depicting the turbidity flow that was measured in his experiments

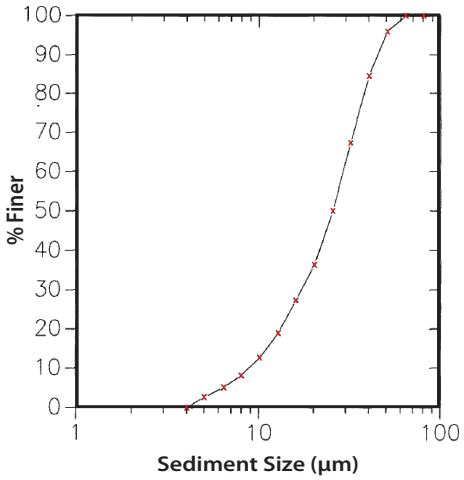


Figure 1.18: Grain size distribution used by García [1994]

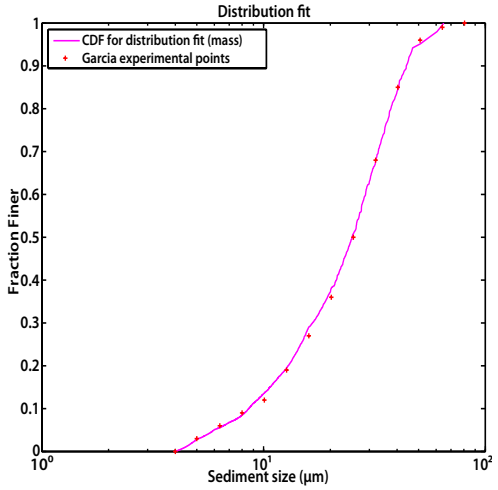


Figure 1.19: Grain size distribution used to compute composite advection lengths

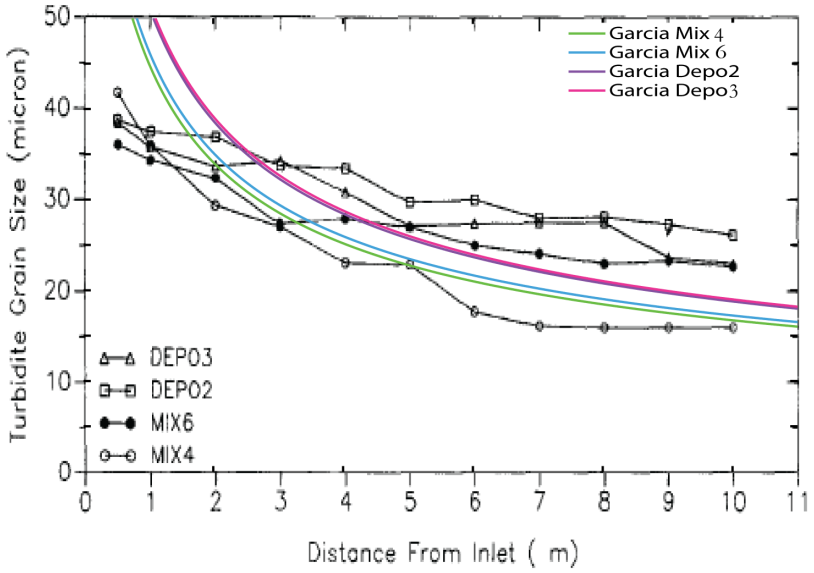


Figure 1.21: Composite advection lengths computed using García's [1994] input parameters and García's [1994] grain size distribution superimposed over the recorded variations of sediment deposit median grain size D_{50} with distance from inlet from García's [1994] experiments

1.1.3 Conclusions

This study has shown that complicated flow (with hydraulic jumps, complex velocity profiles) may create simple deposition profiles, and therefore advection-settling models simplify modeling of depositional turbidity currents. This type of models is accurate for small grain sizes and has already been used successfully by Straub and Mohrig [2008] to model levee growth. Comparing advection length(s) with local scale(s) of topography indicates the need (or not) for more detailed modeling.

Chapter 2

Evolution of the geometry and scales in an Æolian dune field

2.1 Introduction

In the first chapter of this dissertation, we have proposed a way to simplify the computation of deposition patterns for fully depositional turbidity currents. In this chapter, we present a simple code used to study basic statistical information of æolian dune fields; average and standard deviation of dune spacing and dune heights (we call height the length of the vertical altitude) and number of interactions as function of time.

Similar to the advection-settling model presented in chapter 1, this 1D-model focuses more on the global effects of dune interactions on a dune field than on the processes involved in these interactions. While, in the case of fully-depositional turbidity currents, we showed that the topography may have only a second-order effect on the flow – and therefore on its own evolution – , we demonstrate here that in the case of dunes, it is also possible to retrieve some statistical parameters of a dune field by only considering the topography; that is, the flow only has a second-order effect on the evolution of the topography. Therefore, the principle of this simple model is to use an initial topography and evolve this topography using rules based

only on the topography.

2.2 Description of the model

N dunes, numbered from $i = 1$ to $i = N$, are abstracted as similar triangles whose only variables are crest height h_i and crest abscissa x_i; A cartoon representation of such a model is provided in Figure 2.1. The stoss slope angle, which we assume to be 15^o, is called α, while the lee slope angle, which we assume to be 30^o, is called β; these angles are the same for every triangle, and therefore the footprint b_i of dune i is proportional to its height h_i as per equation (2.2.1);

$$b_i = h_i \cdot \left[\frac{1}{\tan(\alpha)} + \frac{1}{\tan(\beta)} \right] \tag{2.2.1}$$

We define the Ripple Index R_{bh} as the base-to-height ratio, i.e.

$$R_{bh} = \frac{1}{\tan(\alpha)} + \frac{1}{\tan(\beta)} \tag{2.2.2}$$

The dune field is composed of triangles aligned along a single line, as one would see in a cross section of a dune field oriented parallel to the primary wind direction. Although this could be a possible extension of the model, we do not consider a second dimension at all, and as a consequence assume that all dunes have the same width and are perfectly aligned. At time $t = 0$, a random distribution of crest heights and an independent distribution of dune spacings (distance between crest heights) are used to initialize the code. I provided a graphical example of a single dune showing internal length relationships in Figure 2.1 and an example of two dunes showing the height and spacing variables on Figure 2.2. The types of

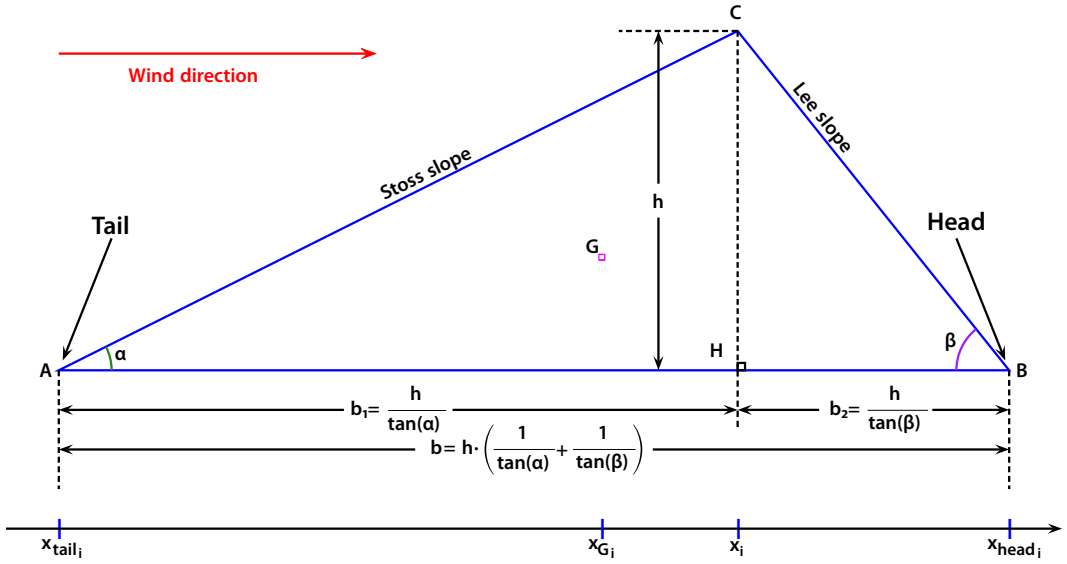


Figure 2.1: Dune model and parameters (not to scale)

distributions – uniform (linear), normal (Gaussian), lognormal – are factors that the user can choose. Boundaries of both the height- and spacing-distributions are fixed and their values are inspired by common values encountered in nature [Kocurek et al., 1992];

$$0.25m \leqslant h_i \leqslant 0.4m \; \forall i \in [\![1;N]\!] \tag{2.2.3}$$

and

$$\mathscr{F}_{sh} \cdot \min(h_i) - R_{bh} \cdot \overline{h_i} \leqslant \Delta x_{h-t} \leqslant \mathscr{F}_{sh} \cdot \max(h_i) - R_{bh} \cdot \overline{h_i} \tag{2.2.4}$$

where Δx_{h-t} is the interdune length, $\overline{h_i}$ is the average height at time 0 and $\mathscr{F}_{sh}$ the height-spacing factor; We use a value $\mathscr{F}_{sh} = 35$ to run the code. For the common case where sand grains majoritarily move as bedload, advection lengths are very small and one may consider that the flow is always at capacity, which means that at all points the entrainment of sediment $w_s \cdot e$ counterbalances the deposition $w_s \cdot c_b$

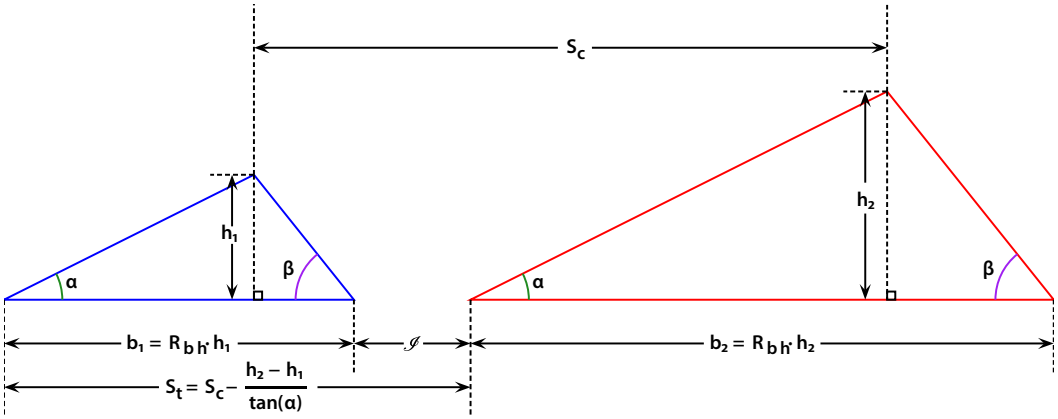


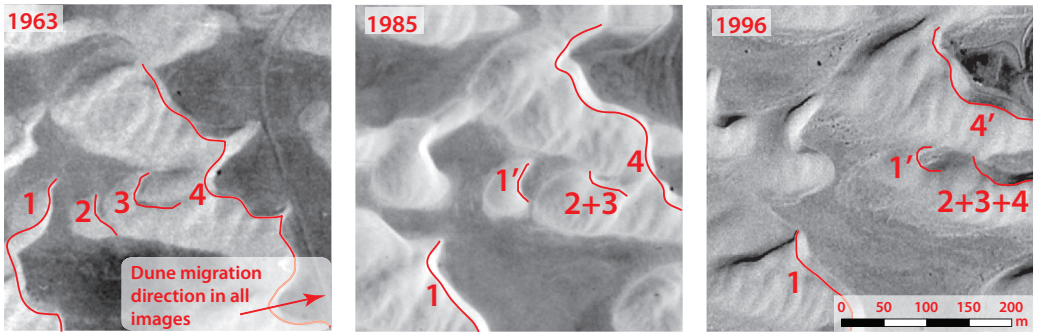
Figure 2.2: Parameters relating two consecutive dunes (not to scale); S_c is the dune spacing; S_t is the tail spacing and is different from the dune spacing for dunes of different heights; $\mathscr{I} = S_c - \left[\frac{h_1}{\tan(\beta)} + \frac{h_2}{\tan(\alpha)}\right]$ is the interdune length. Wind direction is the same as for Figure 2.1.

(using the same notations as in Chapter 1); Consequently,

$$\frac{\partial q_s}{\partial x} = w_s \cdot (e - c_b) = 0 \tag{2.2.5}$$

The spatially averaged sediment flux is therefore considered to be constant at all positions. We also assume that all grains have the same density, so that volume and mass are proportional. At times $t > 0$, these dunes move in the same direction at velocities inversely proportional to their heights (see equation (B.1) in the appendix; an inverse relationship between dune height and velocity is also reported by Endo et al. [2005]). Thus smaller dunes, after a certain interval of time, will catch up with larger, slower-moving dunes, and cause interactions. Interactions may be constructive, neutral or regenerative in terms of pattern development; however, my model only explores two types of interactions: coalescence and repulsion; Examples of these two processes as reconstructed from White Sands data are provided in Fig-

ures 2.3 and 2.4.



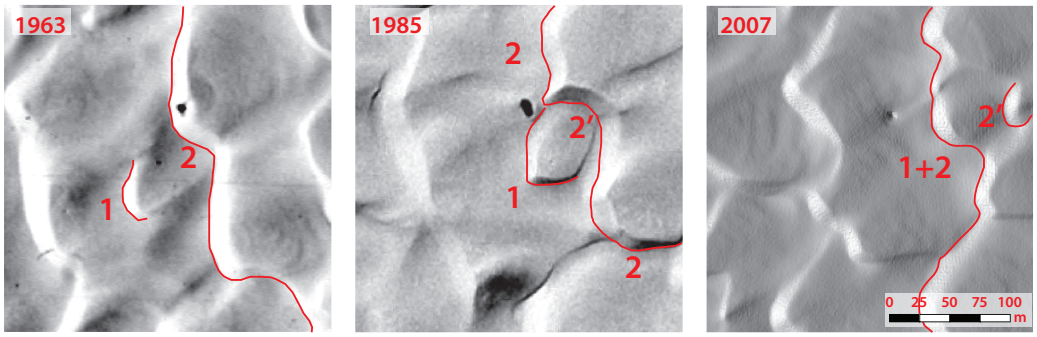
(a) Original configuration of dunes 1 to 4

(b) Configuration after coalescence of dunes 2 and 3

(c) After the combined dune "$2+3$" has broken dune 4 into 4 and $4'$, "$2+3$" coalesces with 4

Figure 2.3: Reconstruction of successive coalescence processes (after Ewing and Kocurek [2010]). Dune 2 of the original configuration (Subfigure (a)) merges successively with Dune 3 (Subfigure (b)) and part of Dune 4 (Subfigure (c)). Subfigure (c) also captures a repulsion that separates 4 and 4'. The red lines outline the crests of the dunes.

The physics of these interactions is still poorly understood, so we have just abstracted the physics as conservation of sediment flux and conservation of mass. When two triangles collide, a set of rules based on the conservation of mass (i.e. the conservation of squared crest heights) and on the conservation of the center of mass transforms the two colliding dunes into either a new single dune (coalescence process) or two new dunes (repulsion process). We assume that collision happens when the tails of two dunes are superimposed, even though the interaction process in nature begins before that moment. However, the discrete abscissæ resulting from discrete time steps may allow a smaller dune to pass over a larger one without an interaction. Therefore, we introduce a tolerance T that is a maximum distance



(a) Original configuration of dunes 1 and 2

(b) Dune 1 begins splitting Dune 2 into "new dune" 2 and 2′

(c) Dune 1 merges with the "new dune" 2 to form "1 + 2" and a smaller Dune 2′ is ejected in front of the "1 + 2"

Figure 2.4: Reconstruction of a repulsion process (after Ewing and Kocurek [2010]). Dune 1 of the original configuration (Subfigure (a)) merges with Dune 2 (Subfigure (b)) while a small part of Dune 2 is ejected and makes Dune 2′ (Subfigure (c)). The red lines outline the crests of the dunes; Dune migration direction is the same as for Figure 2.3.

above which no interactions happen, as described by the "interaction condition" given in equation (2.2.6). Equation (2.2.7) is used for coalescence of upwind dune $\mathcal{A}$ of height $h_{\mathcal{A}}$ with downwind dune $\mathcal{B}$ of height $h_{\mathcal{A}}$, which results in dune $\mathcal{C}$ of height $h_{\mathcal{C}}$, and equation (2.2.8) is used for repulsion between upwind dune $\mathcal{A}$ of height $h_{\mathcal{A}}$ and downwind dune $\mathcal{B}$ of height $h_{\mathcal{B}}$, which results in dunes $\mathcal{D}$ of height $h_{\mathcal{D}}$ (upwind) and $\mathcal{E}$ of height $h_{\mathcal{E}}$ (downwind). In the case of a repulsion, additional rules are needed to have a fully determined process; the first of these rules is purely arbitrary and states that the downwind end ("head") of the new upwind dune touches the upwind end ("tail") of the new downwind dune; the second of these rules uses a predetermined height fraction that is taken off the colliding downwind dune and added to the new upwind dune (i.e. the larger dune gains a fraction p_1 of the height

of the smaller dune during the process). The derivation of the relationships between heights is provided in the appendix, section B.1, and a simplified reason for using the p_1 coefficient of proportionality between the size of the original upwind dune and the size of the final downwind dune in the case of a repulsion is provided in section B.2 of Appendix B.

$$(x_{i+1} - x_i) - \frac{h_{i+1} - h_i}{\tan(\alpha)} \leqslant T \tag{2.2.6}$$

$$h_{\mathcal{C}} = h_{\mathcal{A}} \cdot \sqrt{1 + R_{\mathcal{B}\mathcal{A}}^2}, \quad R_{\mathcal{B}\mathcal{A}} = \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}} \tag{2.2.7}$$

$$\begin{cases} h_{\mathcal{D}} &= h_{\mathcal{A}} \cdot (R_{\mathcal{B}\mathcal{A}} + p_1) \\ h_{\mathcal{E}} &= h_{\mathcal{A}} \cdot \sqrt{1 - p_1^2 - 2 \cdot p_1 \cdot R_{\mathcal{B}\mathcal{A}}} \end{cases} \tag{2.2.8}$$

The ratio of heights of the colliding dunes is used to decide whether coalescence or repulsion happens: dunes with a large difference in size coalesce, while dunes with smaller difference in size repulse each other. We arbitrarily fixed the "maximum merging ratio" (*mmr*) at $mmr = {}^1/_2$: dunes $\mathcal{A}$ (upwind) and $\mathcal{B}$ (downwind) coalesce if and only if $\frac{h_{\mathcal{A}}}{h_{\mathcal{B}}} < \frac{1}{2}$. For perfectly aligned dunes , Katsuki et al. [2005] reports that dunes coalesce when their mass ratio $\frac{h_{\mathcal{A}}^2}{h_{\mathcal{B}}^2}$ is less than or equal to 0.5 and repulse each other if this mass ratio is more than or equal to 0.6.

The new positions of the dunes after an interaction are determined by a conservation of the center of mass; equation (2.2.9) is used for computing the center of mass of Dune $\mathcal{C}$ resulting from the coalescence of Dunes $\mathcal{A}$ and $\mathcal{B}$; equation

(2.2.10) is used for computing the centers of mass of Dunes $\mathcal{D}$ and $\mathcal{E}$ resulting from the repulsion between Dunes $\mathcal{A}$ and $\mathcal{B}$. The derivation is provided in the appendix, section B.3. An example of interaction results is provided in Figure 2.5.

$$x_{\mathcal{C}} = x_{\mathcal{A}} \cdot \left[\frac{1 + R_{\mathcal{B}\mathcal{A}}^2 \cdot s}{r_1} + t \cdot \left(1 + R_{\mathcal{B}\mathcal{A}} - R_{\mathcal{B}\mathcal{A}} \cdot \frac{1 + R_{\mathcal{B}\mathcal{A}}}{r_1} - \sqrt{r_1} \right) \cdot \mathcal{F}_1 \right] \tag{2.2.9}$$

$$\begin{cases} x_{\mathcal{D}} &= \frac{x_{\mathcal{A}}}{r_1} \cdot \left[1 + s \cdot R_{\mathcal{B}\mathcal{A}}^2 + t \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2 \right]^{\frac{3}{2}} \right) \cdot \mathcal{F}_1 + \ldots \right. \\ & \quad \left. \ldots - t \cdot \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)} \right) \cdot \left[r_1 - r_p^2 \right] \right] \\ x_{\mathcal{E}} &= \frac{x_{\mathcal{A}}}{r_1} \cdot \left[1 + s \cdot R_{\mathcal{B}\mathcal{A}}^2 + t \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2 \right]^{\frac{3}{2}} \right) \cdot \mathcal{F}_1 + \ldots \right. \\ & \quad \left. \ldots + t \cdot \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)} \right) \cdot r_p^2 \right] \end{cases} \tag{2.2.10}$$

with

$$\begin{cases} R_{\mathcal{B}\mathcal{A}} &= \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}} \\ r_1 &= 1 + R_{\mathcal{B}\mathcal{A}}^2 \\ r_p &= R_{\mathcal{B}\mathcal{A}} + p_1 \\ s &= \frac{x_{\mathcal{B}}}{x_{\mathcal{A}}} \\ t &= \frac{h_{\mathcal{A}}}{x_{\mathcal{A}}} \\ \mathscr{L} &= \frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)} \\ \mathscr{F}_1 &= \frac{1}{3} \left(-\frac{1}{\tan(\alpha)} + \frac{1}{\tan(\beta)} \right) \end{cases} \tag{2.2.11}$$

At each time step, vectors containing dune heights and positions and the number of remaining dunes are recorded, as well as statistical results such as mean

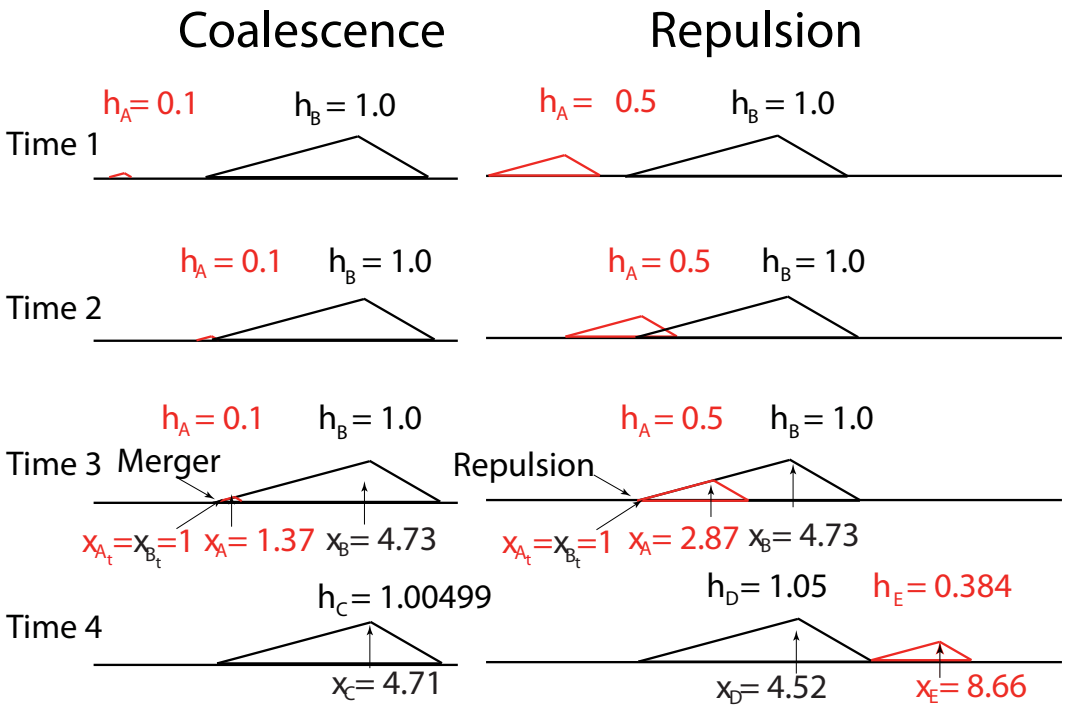


Figure 2.5: Examples of the evolution of dune heights and positions during a coalescence or repulsion process of two dunes. Dune A is the smaller dune, upwind before the interaction; Dune B is the larger dune, downwind before the interaction; Dune C is the result of the coalescence of Dunes A and B on the left side; Dunes D (larger, upwind) and E (smaller, downwind) are the result of the repulsion between Dunes A and B on the right side. Red is used for the smaller dune and black for the larger dune.

height, mean interdune length and spacing, standard deviation of heights, dune spacings and interdune lengths. While interactions in nature may take decades to happen, the model assumes that interactions are instantaneous; After each time step, when dunes have been moved by a distance $d_{i,j} = v_i \cdot \left(t_j - t_{j-1}\right)$ (where $v_{i,j-1}$ is the velocity of dune i between time steps $j-1$ and j, t_{j-1} and t_j the respective times at time steps $j-1$ and j, and $d_{i,j}$ the distance by which dune i moves forward in the

interval of time $\left[t_{j-1};t_j\right]$), the model checks for possibilities of interactions (using equation (2.2.6)) and applies the interaction rules in terms of heights and positions for each of the possible interacting dunes. The new vector of positions and heights replaces that before the interaction; The output file therefore contains exactly as many lines at time steps (including time step 0). Because the velocities of dunes are just a function of dune height, give or take a proportionality constant, the total time is also a function of this proportionality constant. Therefore, only little attention should be paid to the magnitude of the time scale. Also, because coalescence transforms two dunes into a single one (thus reducing the total number of dunes) and repulsion transforms two dunes into two different dunes (thus keeping the same total number of dunes), the total number of dunes as a function of time can only decrease if there is no additional input of sediment; We assume an infinite field (no output-to-input recycling of dunes over a limited distance) and no sediment input. The model is designed to stop either after a given amount of time or when a given number of dunes remain.

This model is deterministic with random input, hence a larger number of interactions reduces the impact of the initial randomness, similarly to the turbidity current model presented in Chapter 1.

2.3 Implementation of the model

The code is implemented in Matlab®as a file to execute by block instead of a function. All variables are therefore "hard-coded" at the beginning of the file. Table C.1 provides the list and definition of these inputs.

Var.: Def.	Value	Type	Length
`theta1`: stoss slope angle in degrees	15	$\mathbb{R}$	1
`theta2`: Lee slope angle in degrees	30	$\mathbb{R}$	1
`t0`: Initial time	0	$\mathbb{R}$	1
`N`: Initial number of dunes	5000	$\mathbb{N}$	1
`minheight`: Minimum height of dunes	0.25	$\mathbb{R}$	1
`maxheight`: Maximum height of dunes	0.4	$\mathbb{R}$	1
`maxvel`: Maximum velocity $v_{\max}$ (in $\mathrm{m}^2 \cdot \mathrm{yr}^{-1}$) in the initial configuration, such that $v_i = v_{\max} \cdot t_{step} \cdot \frac{\min(h_i)}{h_i}$ is the velocity of a dune of height h_i in $\mathrm{m}^2 \cdot {t_{step}}^{-1}$	70	$\mathbb{R}$	1
`hsf`: Height-spacing factor $\mathscr{F}_{sh}$ as defined in equation (2.2.4)	35	$\mathbb{R}$	1
`hgtdist`: Type of distribution of dune heights: 1 for normal, 0 for lognormal, 2 for uniform	1	$\{0,1,2\}$	1
`idldist`: Type of distribution of interdune lengths: 1 for normal, 0 for lognormal, 2 for uniform	1	$\{0,1,2\}$	1
`tmax`: Maximum time in years	500	$\mathbb{R}$	1
`tstep`: Time step t_{step} in years	1/52	$\mathbb{R}$	1
`stopnumber`: If total time not elapsed, code stops when there are fewer than `stopnumber` dunes	3	$\mathbb{N}$	1
`nbins`: Number of bins for histograms	20	$\mathbb{N}$	1
`mmr`: Maximum merge ratio $\frac{h_2-h_1}{h_2}$, if $h_2 = \max(h_1,h_2)$	1/2	$\mathbb{R}$	1
`p1`: When *mmr* is exceeded, fraction of upwind dune height transfered to downwind dune	0.1	$\mathbb{R}$	1
`outputfilehgt`: Name of output file containing the height of each dune at each time step	'heights.txt'	String	N/A
`outputfilepos`: Name of output file containing the position of each dune at each time step	'.positionstxt'	String	N/A
`outputfilesta`: Name of 1st file containing statistics	'statistics.txt'	String	N/A
`outputfilesta2`: Name of 2nd file containing statistics	'statistics2.txt'	String	N/A
`outputfileeve`: Name of file containing list of events (interactions)	'events.txt'	String	N/A
`tolerance`: Size of the tolerance interval when merging dunes	$\min(h_i) \cdot R_{bh} \cdot \frac{2}{100}$	$\mathbb{R}$	1

Table 2.1: List of input variables ; $\mathbb{R}$ is used in the type column for full-precision real variables, while $\mathbb{N}$ is used to integers.

At each time step:

1. We compute the velocity of each dune;
2. We compute the displacement of each dune
3. We compute the tail position of each dune
4. If a smaller dune would "pass over" a larger dune without interaction, we reduce the time step so that this problem does not occur and recompute the displacement of each dune. The resulting time step is not used for all further loops, as this may slow down the code without being necessary.
5. We update the current positions with the values of the former positions plus the displacement
6. From the tail positions, we find location where interactions will happen
7. If the vector containing interaction locations is not empty
   (a) We determine whether a coalescence or a repulsion process happens
   (b) We compute the new positions and heights of the resulting dunes
   (c) The new positions and height replace the old ones in the vectors of positions and heights; in case a coalescence has happened, dunes farther away are moved back one index upwind to compensate for the loss of one dune; The rest of the vectors is filled up with "no value points" (Matlab®'s `NaN`)

(d) The type of interaction is written into the event file, together with the number of remaining dunes

(e) A message is displayed on the screen with the type of interaction, the current time, and the remaining number of dunes

8. We compute statistical variables

9. In case the code has been run with possibility of repulsions, we also compute statistics on the same set of dunes less the dunes smaller (in height) that the mean height minus two standard deviations; The statistics from this variable may be used to assess the effect of smaller dunes "running away", although the spacings are affected by the deletion of smaller dunes

10. We write the height- and position- informations into their respective files

11. We write the statistics into their respective files: first file for the whole field, second file for the field without smaller dunes

12. We update the current matrices of positions and heights so that the current information becomes the old information

13. If the maximum time or minimum number of dunes has been reached, we stop the loop

After the preceding loop has been run, we close all output files and write the position and height information of the last time step into a separate file. We also record at this time the possible necessary adjustments of the time step, and when this happened.

Due to the computer writing output files to the hard drive at each time step (this could be potentially replaced by a buffer accumulating results for a given number of time steps, then writing that buffer into the output file, at the expense of available Random Access Memory), the program is very hard-drive intensive and will only run as fast as the hard drive permits. Also, the output files include full precision variables and therefore require an important amount of space on the hard drive.

2.4 Results

Results were analyzed graphically. Cross-plots allow us to analyze the effects of repeated interactions on the global variables spacings, heights, positions, etc. We provide an example of dune-field modeling with 10 dunes in Figure 2.6. We also give the evolution of the number of dunes as a function of time in Figure 2.8. Some observations can be made using these two plots:

- The field evolves through three main stages (time intervals) linked by transitional stages:

    1. During the first interval, $t = 0$ to $t \simeq 4$, the total number of dunes stays constant. This first interval corresponds to an initial reorganization of the field in terms of dune spacings before any interaction can take place;
    2. During the second interval, $t \simeq 9$ to $t \lessapprox 15$, interactions take place at a fast pace and the number of dune decreases as a power of time. This interval corresponds to a reorganization of the field in terms of dune

heights: smaller dunes are either absorbed by bigger dunes through coalescence or move in front of bigger dunes by repulsion[1];

3. During the third interval, $t \lessapprox 25$ to $t = t_{final}$, interactions still happen, and the number of dunes still follow a power-law of time, but the exponent of that power-law is reduced (divided by 6.5) as the frequency of interactions diminishes strongly; This interval represents a mature dune field, fully organized field, where dune spacing roughly follow a lognormal distribution as shown in Figure 2.7, with similar, smaller spacings between the majority and larger spacings between smaller dunes that are running away.

4. After fitting power-laws in Figure 2.8 and discarding transitional stages, we can put transitions between intervals at $(N,t) = (5000,7)$ and $(N,t) = (328,19.6)$,

- Even though we can differentiate between these three intervals in terms of frequency of interaction, Figure 2.6 shows that interactions take place "by cluster":

  – If larger dunes are close to each other, a repulsion between the first larger dune and a smaller dune will produce another smaller dune, which will them almost immediately interact with the second larger dune; As

[1] In the repulsion process, the smaller dune ejected from the bigger one during the process is not the same as the smaller dune before the process, and "moving in front" as an abuse of language.

dunes reorganize spatially by decreasing height, the conditions for two or more consecutive repulsions become easier to satisfy;

- If larger dunes are not close to each other, this means that they are almost equidistant from each other; as a consequence, smaller dunes (with similar velocities, and similar distances the next larger dune) will reach the next larger dune in comparable times, resulting in almost consecutive interactions.

This behavior is also observed in nature [Ewing, 2008].

Another observation that our model allows is the comparision of the average dune height with the average dune spacing at each time step; Figure 2.9 represents such relationship. Our code shows that whatever the shape of the initial distributions (of heights and spacings), the relationship between average heights and average spacing always follows a power-law, as shown on Figure 2.9. Wasson and Hyde [1983] measured dune heights and spacings of 3 dune fields of the Australian desert and averaged these values for 27 blocks of 10 to 80 dunes for a total of 915 dunes. These values also correlated along two power-law least-square trendlines, as shown on Figure 2.10

2.5 Conclusion

While this simple model is not designed to yield new insight at *why* dunes interact the way they do, we can draw a few conclusions from the good correlations between model behavior and nature:

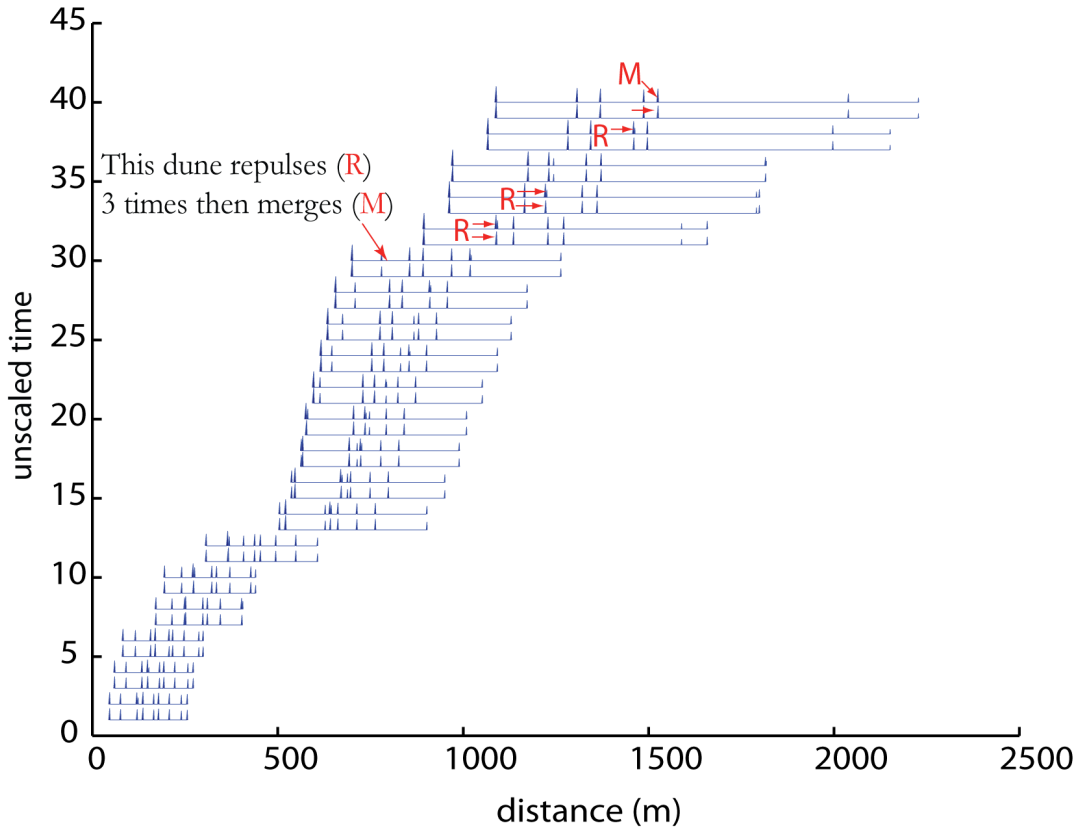


Figure 2.6: Example of interactions in a 10-dune dune field. Reading two lines by two lines, from the bottom-left corner. Each group of two lines represents the dune field before and after interaction. Only times when interactions happen are plotted. Each blue line represents the field at a given time while the small spikes represent the dunes.

- The only part of physics included in this model are the conservation of sediment flux with distance (velocity-height relationship) and the conservation of mass and center of mass during interactions. There is therefore a strong possibility that these two conservation laws are responsible for the organization of dune fields;

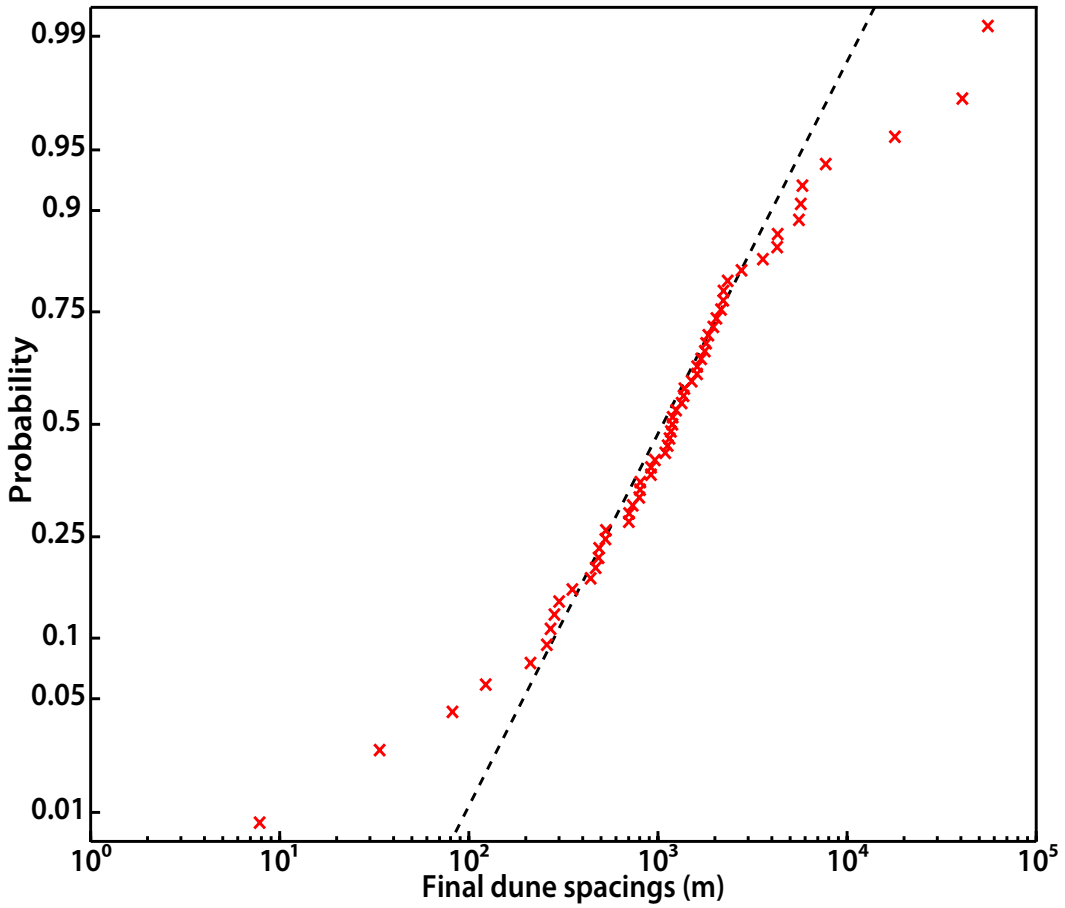


Figure 2.7: Probability plot for a lognormal distribution, applied to the final dune spacings. Initial height- and spacing-distributions were normal. Both coalescence and repulsion were allowed ($mmr = 0.5$).

- The physics of individual interactions needs to be better understood. However, our model shows the advantages of separating full-flow models, which are accurate on a small scale given a fixed computing power, from models that do not consider the flow at all, which provide usable results in terms of statistics at a much larger scale.

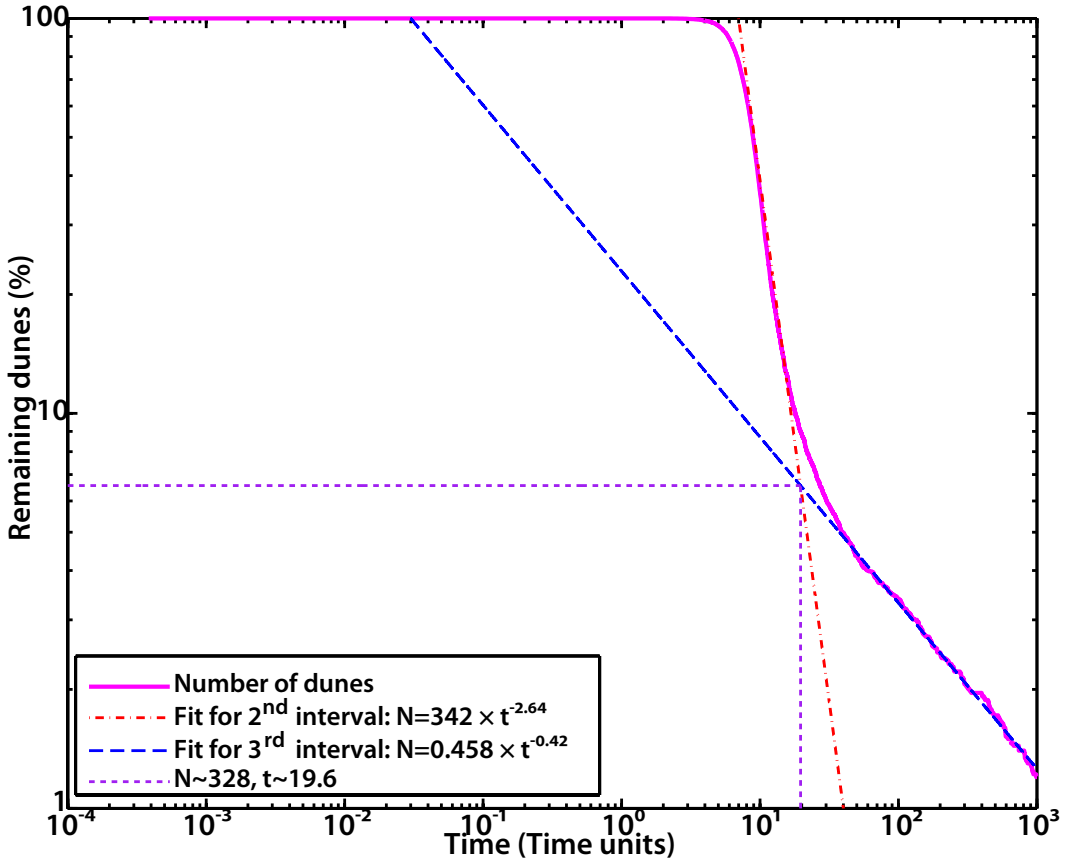


Figure 2.8: Percentage of remaining dunes (with respect to the initial number of dunes) as a function of time, in the case both repulsion and coalescence are allowed by the model ($mmr = 0.5$). Allowing repulsions to happen slightly slows down the decrease of the number of dunes with time, although the same behavior in three time-intervals (stagnation, rapid decrease, slower decrease) are still observed.

- Although our code may not be fully realistic, as it uses an infinite support while dune fields may be restricted in size by the basin dimensions, we believe that the spatial organization of dunes is an indicator of the dune field maturity. The code shows that the distribution of spacings evolves with time towards a lognormal distribution, and therefore comparing the current spac-

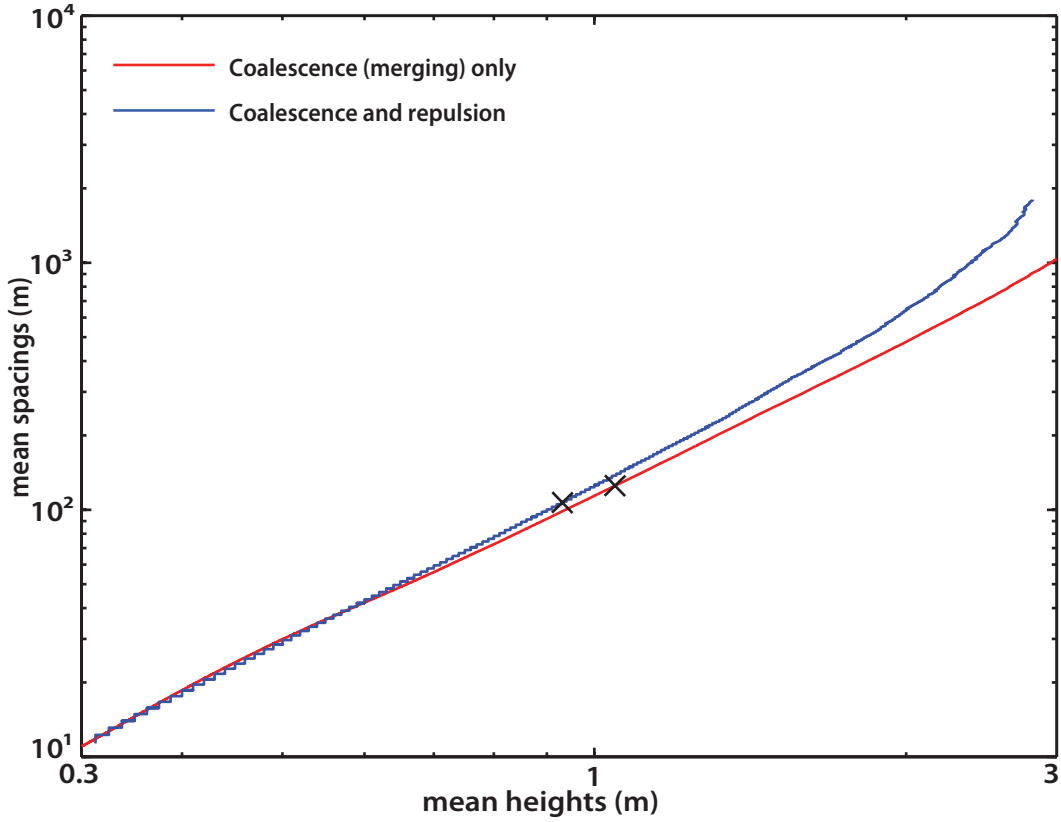


Figure 2.9: Example of average spacing vs average height plot, showing a power-law relationship between both variables. The red line was obtained by running the code with no repulsion ($mmr = 1$) while the blue line is obtained by running the code with both coalescence and repulsion ($mmr = 0.5$). The black crosses on the blue and red lines indicate the time at which only 10% of the initial number of dunes remain. The departure between both lines at larger mean heights may be due to the smaller dunes "running away´´ in downwind of the larger dunes.

ing distribution of a dune field to a lognormal distribution may yield an indication of how mature the field is.

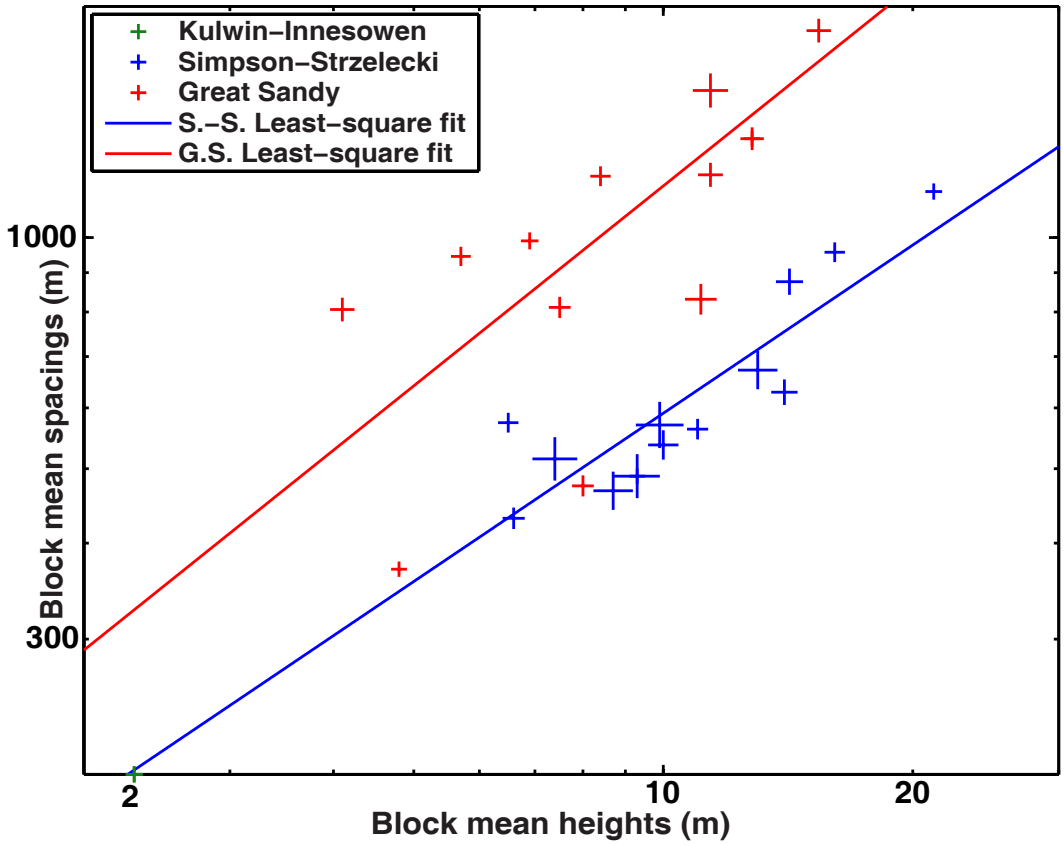


Figure 2.10: Average dune spacing vs average dune height as shown by Wasson and Hyde [1983]. Averages were computed in 27 blocks of 3 dune fields of the Australian desert. The size of each cross is proportional to the number of dunes in each block, from 10 to 80. Lines are best-fit lines for the two dune fields where more than 1 block was analyzed.

Chapter 3

Relationships between geometrical parameters of rivers in different planes

Motivation

The advent of modern computing and the availability of large datasets gives us an occasion to revisit "soft relationships" between geometric parameters of channels that have been hypothesized or demonstrated manually. Manual computations of the radius of curvature, for instance by fitting circles, is inaccurate, not reproducible, and time consuming, and therefore does not allow one to obtain a sufficiently large number of values to make statistics. We present in this chapter a Matlab© code that computes the local radius of curvature and width of a channel in plan form. River coordinates are smoothed to reduce scatter due to digitization, and a second-order, finite-difference code computes the local radius of curvature. The width of the channel, defined as the distance between the left and right banks perpendicularly to the centerline, is then computed using a linear interpolation between bank datapoints. We applied this code successfully to plan-form data from the Mississippi and Trinity rivers. We used Mississippi River data to study the relationships between radius of curvature and water depth, bed composition and cross-sectional shape; We analyzed Trinity River data to obtain width data and possibly migration rate as functions of radius of curvature. The comparison of the behavior

of both river systems allowed us to make preliminary observations on how to differentiate rapidly-migrating channels from slowly-migrating channels, or "actively meandering channels" from "stable meandering channels" as defined by Lagasse et al. [2004].

3.1 Introduction

River meandering has been for many years an object of study for geologists and geophysicists. Bagnold [1960] and Leopold and Wolman [1960] related the shape of river meanders (mean radius of curvature/channel width) to flow resistance. Langbein and Leopold [1966] derived a theoretical equation for the ideal shape of a meander, based on work by von Schelling [1964]:

$$\phi(x) = \omega \cdot \sin\left(\frac{2 \cdot \pi \cdot x}{M}\right) \tag{3.1.1}$$

where ϕ is the local angle between the channel and the mean downstream direction, ω is the largest value of ϕ (angle between the channel tangent and the mean downstream direction when the channel crosses the line of mean downstream direction), x is the distance along the channel centerline, and M is the distance along the channel that corresponds to a spatial period. Gedzelman [1974] matched this ideal shape for meanders to meanders in the Gulf Stream. Map view patterns are related to the dynamics of rivers [Ferguson et al., 2003, Corney et al., 2006] and the way these rivers evolve in time, hence the need to be able to extract the basic geometrical parameters of a natural channel (curvature or radius of curvature,

width). While existing programs that compute these parameters are often cumbersome and act like black boxes to many users, the code presented in this chapter is designed to provide the geoscience community with another tool that does not suffer from these deficiencies. This code, written in Matlab©, is designed for geoscientists who need more accurate and reproducible values of channel width and curvature than what manual measurements may yield. We evaluated using three methods: visually using specific geometric forms with known loci of centers of curvature, numerically using idealized channel forms, and with natural channels. These evaluations quantify the potential error and pitfalls of the developed methods. Data from the Mississippi River in Louisiana and the Trinity River is Texas are then presented to quantitatively evaluate the connection between channel curvature and cross-sectional geometry, as well as channel migration rate. Such relationships have been qualitatively and semi-quantitatively proposed in the past. Here we seek to analyze sufficient data to quantify these correlations and provide a framework for connecting them to river dynamics.

3.2 Mathematical description and code implementation

The "Curvature and width" code is composed of several modules or functions that are then used consecutively by the main code: a simple filtering function that reduces the scatter due to manual digitization of the river centerline, a curvature function that computes the radius of curvature of the filtered centerline as well as the locations of the centers of curvature of the filtered centerline, and a width function that computes the width of the channel. We will describe the functions

separately, beginning with the curvature function.

3.2.1 Implementation of the curvature code

The curvature of a plane curve $\gamma(x)$ is defined as the norm of the derivative of $\overrightarrow{T}(x)$ with respect to x, with x the distance along the curve and $\overrightarrow{T}(x)$ the normalized tangent vector to the curve. The radius of curvature is the inverse of the curvature and is more significant because this variable has the unit of a length. This definition, however, is only useful to geoscientists in a few cases. One of these cases is deriving the curvature of the channels whose plan-form shape may be described by equation (3.1.1), or of other channels that may be described by an equation that is a function of the distance along these channels (which we call the curvilinear length of the curve). Another case is deriving an equation for the curvature of a channel parametrically defined by its coordinates (where t is the time at a given point, if we were to travel along the channel at a constant speed), which is the standard format of digitized data from natural channels. The radius of curvature for a channel parameterized by $(x[t], y[t])$ is given by equation (3.2.1):

$$R(t) = \frac{\left[\left(\frac{\partial x}{\partial t}\right)^2 + \left(\frac{\partial y}{\partial t}\right)^2\right]^{\frac{3}{2}}}{\frac{\partial x}{\partial t} \cdot \frac{\partial^2 y}{\partial t^2} - \frac{\partial^2 x}{\partial t^2} \cdot \frac{\partial y}{\partial t}} \tag{3.2.1}$$

Two different finite-difference schemes may be used, and two different functions allow the user to choose his preferred scheme. In the first function, called `Curvature.m`, we assume that $\partial t = 1$, and we use a centered-difference to express the derivatives of x and y with respect to t; Given the coordinates of a digitized

channel $(x[t], y[t])$, we use the following approximation (equation (3.2.2)):

$$\begin{array}{ll} \frac{\partial x}{\partial t}[i] \simeq \frac{x[i+1]-x[i-1]}{2}; & \frac{\partial^2 x}{\partial t^2}[i] \simeq x[i+1]-2\cdot x[i]+x[i-1] \\ \frac{\partial y}{\partial t}[i] \simeq \frac{y[i+1]-y[i-1]}{2}; & \frac{\partial^2 y}{\partial t^2}[i] \simeq y[i+1]-2\cdot y[i]+y[i-1] \end{array} \tag{3.2.2}$$

Which results in equation (3.2.3):

$$R[i] = -\frac{1}{8}\cdot \frac{\left[(x[i+1]-x[i-1])^2+(y[i+1]-y[i-1])^2\right]^{\frac{3}{2}}}{x[i-1]\cdot(y[i+1]-y[i])-x[i]\cdot(y[i+1]-y[i-1])+x[i+1]\cdot(y[i]-y[i-1])} \tag{3.2.3}$$

The centers of curvature of the channel, which are located on the normal to the channel at a distance of R from the channel, are then obtained with (equation (3.2.4)):

$$\begin{array}{l} C_x[i] = x[i] - \frac{2\cdot R[i]\cdot(y[i+1]-2\cdot y[i]+y[i-1])}{\sqrt{(x[i+1]-x[i-1])^2+(y[i+1]-y[i-1])^2}} \\ C_y[i] = y[i] + \frac{2\cdot R[i]\cdot(x[i+1]-2\cdot x[i]+x[i-1])}{\sqrt{(x[i+1]-x[i-1])^2+(y[i+1]-y[i-1])^2}} \end{array} \tag{3.2.4}$$

In the second curvature function, called `Courbure.m`, we assume that the (x,y) coordinates of the river centerline are those of a function $(x, y(x))$, and we can obtain the local radius of curvature using equation (3.2.5):

$$R(x,y(x)) = \frac{\left[1+\left(\frac{\partial y}{\partial x}\right)^2\right]^{\frac{3}{2}}}{\frac{\partial^2 y}{\partial x^2}} \tag{3.2.5}$$

The finite-difference code then assumes that:

$$\left.\begin{aligned} \frac{\partial y}{\partial x}[i] &= \frac{1}{2}\cdot\left(\frac{y[i+1]-y[i]}{x[i+1]-x[i]}+\frac{y[i]-y[i-1]}{x[i]-x[i-1]}\right) \\ \frac{\partial^2 y}{\partial x^2}[i] &= \frac{\dfrac{y[i+1]-y[i]}{x[i+1]-x[i]}-\dfrac{y[i]-y[i-1]}{x[i]-x[i-1]}}{\dfrac{1}{2}\cdot(x[i+1]-x[i-1])} \end{aligned}\right\} R[i] = \frac{\left[1+\left(\dfrac{\partial y}{\partial x}[i]\right)^2\right]^{\frac{3}{2}}}{\dfrac{\partial^2 y}{\partial x^2}[i]} \tag{3.2.6}$$

$$C_x[i] = x[i] - \frac{R[i]\cdot\dfrac{\partial y}{\partial x}[i]}{\sqrt{1+\left(\dfrac{\partial y}{\partial x}[i]\right)^2}} \tag{3.2.7}$$

$$C_y[i] = y[i] + \frac{R[i]}{\sqrt{1+\left(\dfrac{\partial y}{\partial x}[i]\right)^2}} \tag{3.2.8}$$

The locus of the centers of curvature of a curve is called the evolute of this curve. Both codes were tested on mathematical curves with known evolutes (see paragraph 3.3.1). While the theory indicates that the curve should be traveled over at a constant velocity, the assumption used in the first code that the curve is traveled over with a constant time step does not induce much error as long as the digitization is regular in space. With a slightly irregular digitization in space, the error remains generally low, with the notable exceptions of cusps points of the evolute, and some particular curves. The second code assumes that we have a function $(x, y(x))$ and therefore can come into trouble with vertical tangents: this is handled by a second

run on a 45^o-rotated dataset that is seamlessly merged with the results of the function; some issues may however remain (see section 3.3.2). Generally, as shown in Figures 3.1 to 3.12, it appears that the first code handles smooth mathematical curves better than the second code (which fails on the logarithmic spiral); however, the second code handles numerical error better than the first one.

3.2.2 Implementation of the smoothing/filtering algorithm

The smoothing algorithm filters the centerline coordinates to reduce the scatter resulting from the digitization of these data. Two types of filtering windows are combined: a Chebyshev window and a Ricker ("Mexican hat") window. Harris [1978] defines the Chebyshev window of length $2N-1$ and side-lobe attenuation parameter α (decimal logarithm of ratio of sidelobe over main lobe, in the frequency domain) as the Discrete Fourier Transform $w_{\text{Cheb}}(N)$ of the frequency-domain function $W(k)$ given by equation (3.2.9):

$$\begin{cases} W(k) &= (-1)^n \cdot \dfrac{\cos\left(N \cdot \arccos\left[\beta \cdot \cos\left(\pi \cdot \dfrac{k}{N}\right)\right]\right)}{\cosh\left[N \cdot \arccos\left(\beta\right)\right]}, \ \forall\, 0 \leq |k| \leq N-1 \\ \beta &= \cosh\left[\dfrac{1}{N} \cdot \ \text{arccosh}\left(10^{\alpha}\right)\right] \end{cases} \tag{3.2.9}$$

$$\arccos(X) = \begin{cases} \dfrac{\pi}{2} - \arctan\left(\dfrac{X}{\sqrt{1-X^2}}\right) & |X| < 1 \\ \ln\left(X + \sqrt{X^2-1}\right) & |X| \geq 1 \end{cases} \tag{3.2.10}$$

The Chebyshev window produced by Matlab©'s signal processing toolbox has a default value of $\alpha = 5$, which we do not modify. We define a Ricker window $R(M)$ of length M by equation (3.2.11):

$$w(i) = \begin{cases} 1 & M = 1 \\ \left[1 - \left(1 + \dfrac{M-1}{4}\right) \cdot i^2\right] \cdot \exp\left[-(\pi \cdot i)^2\right] & M > 1 \end{cases} \qquad \forall\, i \in \left\{-1, -1 + \frac{2}{M-1}, \ldots, 1 - \frac{2}{M-1}, 1\right\} \tag{3.2.11}$$

$$R(i) = \frac{w(i)}{\sum_i w(i)} \tag{3.2.12}$$

The Chebyshev window works best on long stretches of almost linear channel (large radii of curvature), as using this window in tight bends will result in the filtered centerline cutting through the bend. Conversely, the Ricker window works best on parts of rivers that exhibit small radii of curvature (tight bends) but does not reduce the digitization-induced scattering as much as the Chebyshev window. Therefore, the user is asked to input a parameter θ, so that the centerline coordinates are filtered by a composite window $W_C(L)$ (L being an odd number) that combines a Chebyshev window of length L with a Ricker window of length $\frac{L+1}{2}$ (padded with zeros on both sides so the maxima of both windows coincide). This composite window is defined by equation (3.2.13):

$$W_C = \theta \cdot R\left(\frac{L+1}{2}\right) + (1-\theta) \cdot W_{Cheb}(L) \tag{3.2.13}$$

The filtered coordinates are then equal to the weighted average of the original centerline coordinates, the weights being the coefficients of the composite window. Because filtering the centerline coordinates by a composite window of length L requires that a given point of index i be transformed into the weighted average of neighboring $i - \frac{L-1}{2}$ to $i + \frac{L-1}{2}$, the first $\frac{L-1}{2}$ of the centerline are filtered with windows of increasing sizes, while the last $\frac{L-1}{2}$ points of the centerline are filtered with windows of decreasing sizes.

We found experimentally that $\theta = 0.6$ works well for the lowermost part of the Mississippi River, while $\theta = 1$ (Ricker wavelet only) works best for the lower Trinity River (see example section).

3.2.3 Implementation of the width function

We chose to define the width with respect to the centerline, such that the width of channel at a given centerline point is the length of the segment normal to the centerline at that point and whose extremities belong to each bank of the channel. This definition works well when both banks are subparallel to each other and to the centerline. In other cases, defining a river width does not have much physical meaning, and, in case our code encounters these cases, these points are left in the width vector as "no value points" (Matlab©'s `NaN`).

The first step we use to find the river width associated with a given point in the centerline is to calculate the distances $d_{ij} = P_C[i]\,P_L[j]$ and $d_{ik} = P_C[i]\,P_R[k]$ between each centerline point ($P_C[i]$) and each point of each bank ($P_L[j]$ and $P_R[k]$, on the left and right bank respectively), and find the points on each bank ($P_L[i]$ and

$P_R[i]$) that minimize these distances: $P_C[i]P_L[i] = \min_{j\in[\![1;N_L]\!]} d_{ij}$ and $P_C[i]P_R[i] = \min_{k\in[\![1;N_R]\!]} d_{ik}$. We assume that the bank points belonging to the centerline normal at $P_C[i]$, which we call $N_L[i]$ and $N_R[i]$, are close to the $P_L[i]$ and $P_R[i]$; the user does indeed input a "Swipe length" S, and the code assumes that that $N_L[i]$ is located between bank points $P_L\left[i+\frac{S-1}{2}\right]$ and $P_L\left[i-\frac{S-1}{2}\right]$ (respectively, $N_R[i]$ is located between bank points $P_R\left[i+\frac{S-1}{2}\right]$ and $P_R\left[i-\frac{S-1}{2}\right]$. Assuming that the centerline has been digitized into M_C points, the left and right banks into M_L and M_R points respectively, two matrices of sizes $M_C \times M_L$ and $M_C \times M_R$ are created; the code subsequently looks for the minimum of each matrix line and adds the corresponding index into a vector. The user may want to change the vectorized part of this process into a loop in case an "out-of-memory" is encountered.

In a second step, for each centerline point $P_C[i]$, we compute the dot products $D_{N,\star}[i,k]$ and $D_{T,\star}[i,k]$ of the normalized centerline-normal vector $\overrightarrow{N}[i]$ and normalized centerline-tangent vector $\overrightarrow{T}[i]$, with each normalized vector $\overrightarrow{P_CP_L}[k]$ and $\overrightarrow{P_CP_R}[k]$, for all $k \in \left\{i-\frac{S-1}{2},\ldots,i,\ldots,i+\frac{S-1}{2}\right\}$: $D_{N,\star}[i,k] = \overrightarrow{N}[i]\cdot\overrightarrow{P_CP_\star}[k]$ and $D_{T,\star}[i,k] = \overrightarrow{T}[i]\cdot\overrightarrow{P_CP_\star}[k]$, $\star = R$ or L. The code then first looks for the k indices corresponding to values of $D_{N,\star}[i,k]$ equal to 1 or -1 (with a 10^{-4} tolerance due to numerical error). If such points are found, then we know that an existing bank point is exactly on the centerline-normal. If no such point is found, the code looks the maximum values of $\frac{1}{\left|1-\left|D_{N,\star}[i,k]\right|\right|}$ (called $k_{\max,L}$ and $k_{\max,R}$, or $k_{\max,\star}$ when talking indiscriminately about the left or right bank) and assumes that the centerline-normal is located either between $k_{\max,\star}$ and $k_{\max,\star}+1$, or between $k_{\max,\star}$ and $k_{\max,\star}-1$. The function $x \mapsto \frac{1}{|1-|x||}$ helps finding values that are the closest to 1 or -1, which

are asymptotes of the function. Once $k_{\max,\star}$ is found, the code compares the signs of $D_{T,\star}[i,k]$, $k \in \{k_{\max,\star}-1, k_{\max,\star}, k_{\max,\star}+1\}$ to find where (on the left or on the right of $k_{\max,\star}$) signs change. We call this point $k_{\text{sign},\star}$. Using a linear interpolation between bank points and the values of the dot products $D_{T,\star}[i,k_{\max,\star}]$ and $D_{T,\star}\left[i,k_{\text{sign},\star}\right]$, we compute the coordinates of the points at the intersection of the centerline-normal and the banks, $N_\star[i]$, with equation (3.2.14):

$$(N_\star[i])_x = (P_C[i])_x + F_\star \cdot \left(\vec{N}[i]\right)_x \tag{3.2.14}$$

$$(N_\star[i])_y = (P_C[i])_y + F_\star \cdot \left(\vec{N}[i]\right)_y \tag{3.2.15}$$

$$P_c N_\star[i] = |F_\star| \tag{3.2.16}$$

$$F_{0,\star} = \frac{\left[(P_C[i])_x \cdot \left(\vec{N}[i]\right)_y - (P_C[i])_y \cdot \left(\vec{N}[i]\right)_x\right] - \left[\left(P_\star\left[k_{sign,\star}\right]\right)_x \cdot \left(\vec{N}[i]\right)_y - \left(P_\star\left[k_{sign,\star}\right]\right)_y \cdot \left(\vec{N}[i]\right)_x\right]}{\left(\vec{N}[i]\right)_y \cdot \left[(P_\star[k_{\max,\star}])_x - \left(P_\star\left[k_{sign,\star}\right]\right)_x\right] - \left(\vec{N}[i]\right)_x \cdot \left[(P_\star[k_{\max,\star}])_y - \left(P_\star\left[k_{sign,\star}\right]\right)_y\right]} \tag{3.2.17}$$

$$F_\star = \frac{F_{0,\star} \cdot \left[(P_\star[k_{\max,\star}])_x - (P_\star[k_{sign,\star}])_x\right] + \left[(P_\star[k_{sign,\star}])_x - (P_C[i])_x\right]}{\left(\vec{N}[i]\right)_x} \tag{3.2.18}$$

Where the subscript x or y next to a symbol indicates the x or y coordinate of the point or vector designated by that symbol, $\overrightarrow{N}[i]$ is the normalized centerline-normal vector (obtained from the output of the curvature code), $N_\star[i] =$

$N_L[i]$ or $N_R[i]$ is the intersection of the left (respectively right) bank with the centerline-normal at point $P_C[i]$, and $P_\star\left[k_{\text{sign},\star}\right] = P_L\left[k_{\text{sign},L}\right]$ or $P_R\left[k_{\text{sign},R}\right]$ and $P_\star[k_{\max,\star}] = P_L[k_{\max,L}]$ or $P_R[k_{\max,R}]$ are the two points belonging to the digitized left (respectively right) bank that surround $N_\star[i]$.

If for $k \in \{k_{\max,\star}-1, k_{\max,\star}, k_{\max,\star}+1\}$ the code cannot find the change in signs in $D_{T,\star}[i,k]$, it assumes that somehow the center-point $k_{\max,\star}$ was misplaced and tries to find the sign change in $D_{T,\star}[i,k]$ among the other points included in the swipe length, then repeats the same process as above to determine the exact position of $N_\star[i]$. Usually, a few more values may be determined through that procedure, and a "No value" point (`NaN`) is added to the list of $N_\star[i]$ where a width value cannot be determined at this stage.

Because of our definition of channel width, and because of the assumption that, given a centerline point, the intersection of the centerline-normal with the banks is located within a user-defined number of bank points that minimize the distance from the banks to that centerline point, problems may arise when banks are not sub-parallel to the channel centerline and to each other. "No value" points are the result of the procedure described above. However, in an attempt to solve such issues, our program runs a second pass of the same procedure, with an increased swipe length, and just for the problematic points. Additionally, if all above processes fail (which may be the case at the extremities of the datasets), an angle between 80^o and 90^o between a vector $\overrightarrow{P_C P_\star}[k]$ and the tangent to the centerline will be considered close enough to the normal ("lax condition"), and the coordinates of any point $P_\star[k]$ such that $80^o \leq \widehat{\overrightarrow{P_C P_\star}[k], \overrightarrow{T}} \leq 90^o$ will be added to the list of $N_\star[i]$.

If nothing at all can be found to satisfy any of these criteria, then a "No value" point (`NaN`) is added to the list of $N_\star[i]$.

Theoretically, both bank points belonging to the centerline-normal at a given centerline point should be perfectly aligned with the centerline point. However, due to numerical error and, at the end of datasets, the "lax condition", this is not always the case. We therefore compute two values of the channel width for each centerline point: the first one equal to the sum of the distances between the centerline point and the bank points, $W_C = P_C N_L + P_C N_R$, the second one equal to the "direct" distance between the bank points, $W_D = N_L N_R$. Finally, the width function outputs two accuracy estimates A_1 and A_2; A_1 compares the dot products $\overrightarrow{P_C N_\star[i]} \cdot \overrightarrow{N[i]}$ to 1, independently for each bank, and A_2 compares W_C to W_D.

$$A_1 = 1 - \frac{\sum_{i=1}^{M_C} \overrightarrow{P_C N_\star[i]} \cdot \overrightarrow{N[i]}}{M_C} \tag{3.2.19}$$

$$A_2 = \frac{\text{Card}\left\{ i,\ \left| \frac{W_C[i]}{W_D[i]} \right| < 0.99,\ i \in [\![1; M_C]\!] \right\}}{M_C} \tag{3.2.20}$$

Both accuracy estimates are output as percentages. A_1 and A_2 close to 0% indicate a good accuracy of the width function. A plot of the channel with drawn width segments (centerline to bank 1, centerline to bank 2), available in the example files, may highlight what appears as problem (segments not aligned), although this is often the result of non-filtered center-points being used for this plot when filtered points are used for the computations, while keeping the same bank points as width-

segment extremities. These accuracy estimates, however, are also based on the centerline-normal vector obtained from the curvature function, and therefore do not take into account any error that may be introduced by the finite-difference scheme used in that function. Details of the implementation, including instructions for using the code, are provided in C.1.

3.3 Examples

3.3.1 Mathematical figures

We created a few mathematical figures with known evolutes to test our curvature codes in a purely visual way. These figures include an astroid (equation (3.3.1)), a deltoid (equation (3.3.2)), a tractrix (equation (3.3.3)), a logarithmic spiral (equation (3.3.4)), a cycloid (equation (3.3.5)) and a parabola (equation (3.3.6)).

3.3.1.1 Astroid

$$\begin{cases} x = \cos^3(\theta) \\ y = \sin^3(\theta) \end{cases}, \ \theta \in [0; 2 \cdot \pi] \tag{3.3.1}$$

The evolute of an astroid is a larger astroid, rotated 45^{o} from the original and whose branch centers coincide with the cusp points of the original astroid. Results of functions `Courbure.m` and `Curvature.m` are given in Figures 3.1 and 3.2 respectively.

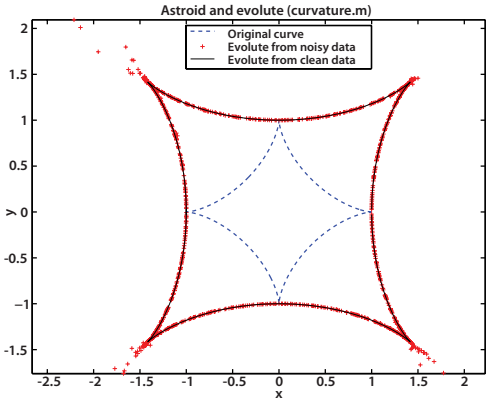


Figure 3.1: Astroid and evolutes computed with code `Curvature.m`. The dashed blue line is the original astroid; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

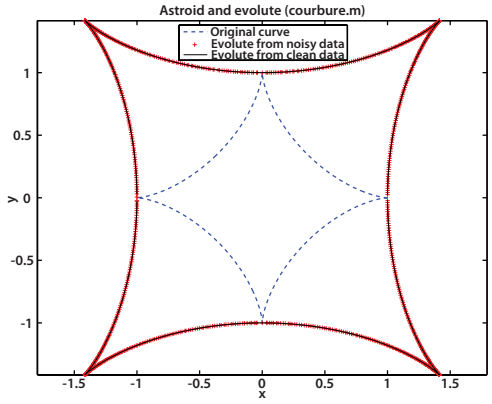


Figure 3.2: Astroid and evolutes computed with code `Courbure.m`. The dashed blue line is the original astroid; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

3.3.1.2 Deltoid

$$\begin{cases} x = 2 \cdot \cos(\theta) + \cos(2 \cdot \theta) \\ y = 2 \cdot \sin(\theta) - \sin(2 \cdot \theta)\, s \end{cases}, \ \theta \in [0; 2 \cdot \pi] \tag{3.3.2}$$

Similarly, the evolute of a deltoid is a larger deltoid, rotated 60^{o} from the original and whose branch centers coincide with the cusp points of the original deltoid. Results of functions `Courbure.m` and `Curvature.m` are given in Figures 3.3 and 3.4 respectively.

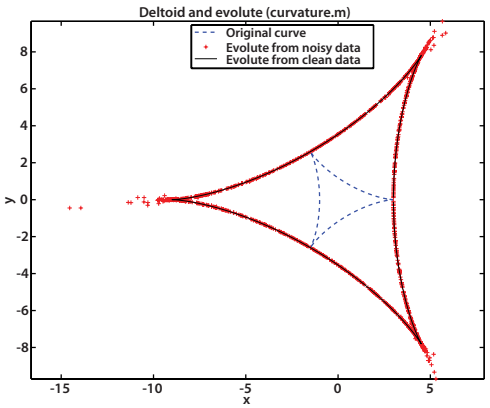


Figure 3.3: Deltoid and evolutes computed with code `Curvature.m`. The dashed blue line is the original deltoid; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

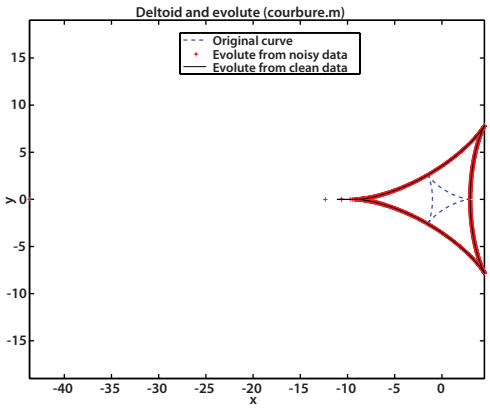


Figure 3.4: Deltoid and evolutes computed with code `Courbure.m`. The dashed blue line is the original deltoid; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

3.3.1.3 Tractrix

$$\begin{cases} x = 4 \cdot [\tanh(u) - u] \\ y = \dfrac{4}{\cosh(u)} \end{cases}, \; u \in [-10; 10] \tag{3.3.3}$$

The evolute of a tractrix is a catenary that touches the tractrix at its cusp point. Results of functions `Courbure.m` and `Curvature.m` are given in Figures 3.5 and 3.6 respectively.

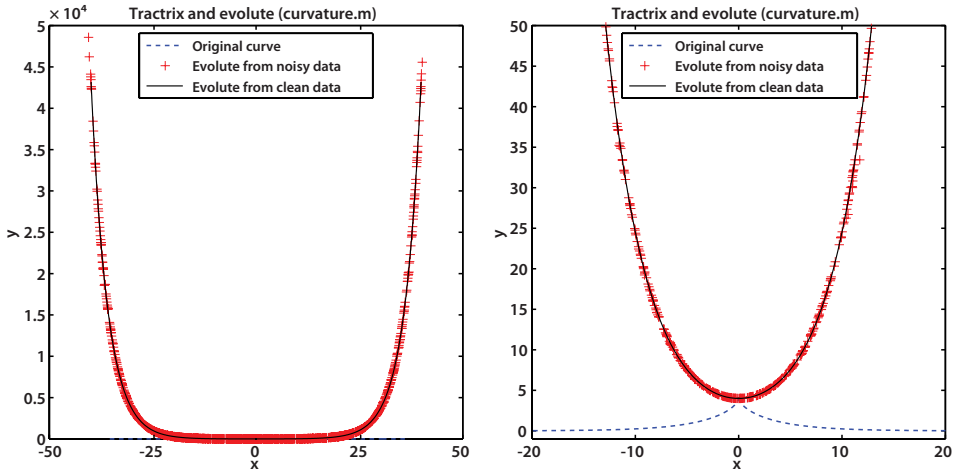


Figure 3.5: Tractrix and evolutes computed with code `Curvature.m`. The dashed blue line is the original tractrix; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data. Both figures represent the same data but have different scales.

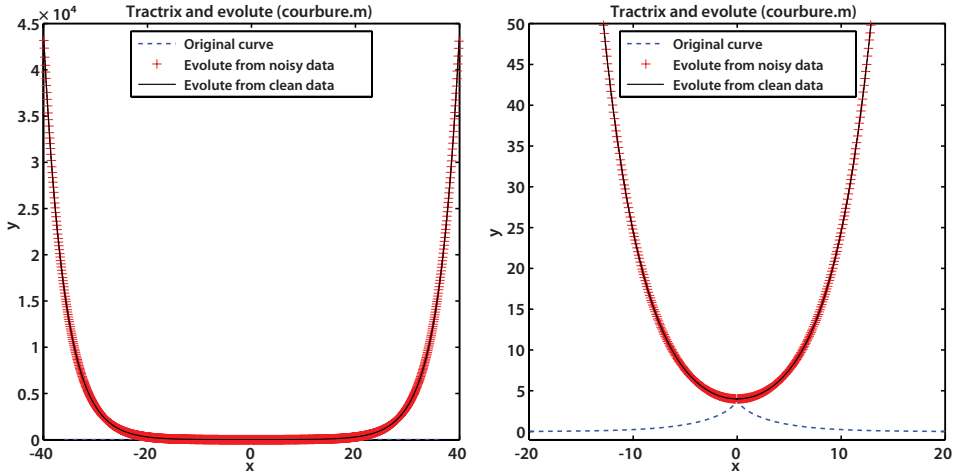


Figure 3.6: Tractrix and evolutes computed with code `Courbure.m`. The dashed blue line is the original tractrix; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data. Both figures represent the same data but have different scales.

3.3.1.4 Logarithmic spiral

$$\begin{cases} x = 1 \cdot \exp\left(\dfrac{\theta}{10}\right) \cdot \cos(\theta) \\ y = 1 \cdot \exp\left(\dfrac{\theta}{10}\right) \cdot \sin(\theta) \end{cases}, \ \theta \in [0; 20 \cdot \pi] \tag{3.3.4}$$

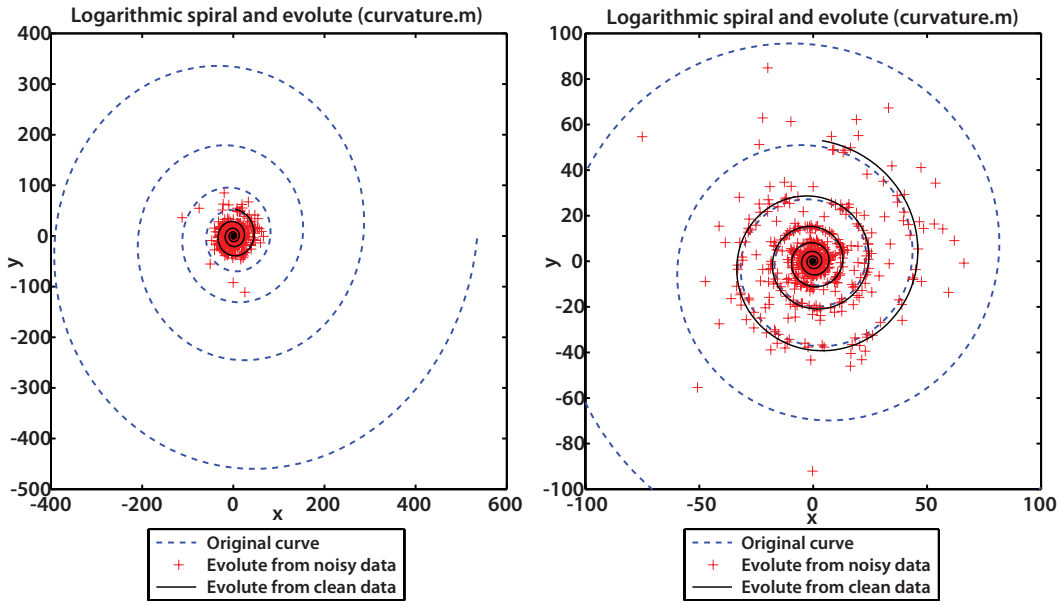


Figure 3.7: Logarithmic spiral and evolutes computed with code `Curvature.m`. The dashed blue line is the original logarithmic spiral; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data. Both figures represent the same data but have different scales.

The evolute of a logarithmic spiral is a congruent logarithmic spiral. Results of functions `Courbure.m` and `Curvature.m` are given in Figures 3.7 and 3.8 respectively.

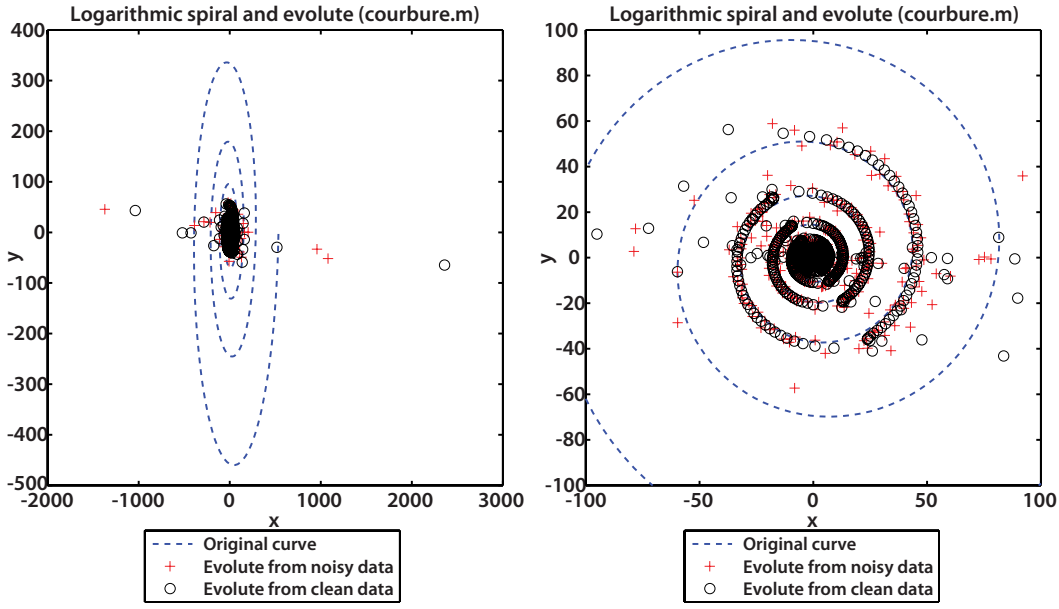


Figure 3.8: Logarithmic spiral and evolutes computed with code `Courbure.m`. The dashed blue line is the original logarithmic spiral; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data. Both figures represent the same data but have different scales.

3.3.1.5 Cycloid

$$\begin{cases} x = \dfrac{\theta - \sin(\theta)}{2} \\ y = \dfrac{1 - \cos(\theta)}{2} \end{cases}, \ \theta \in [0; 6 \cdot \pi] \tag{3.3.5}$$

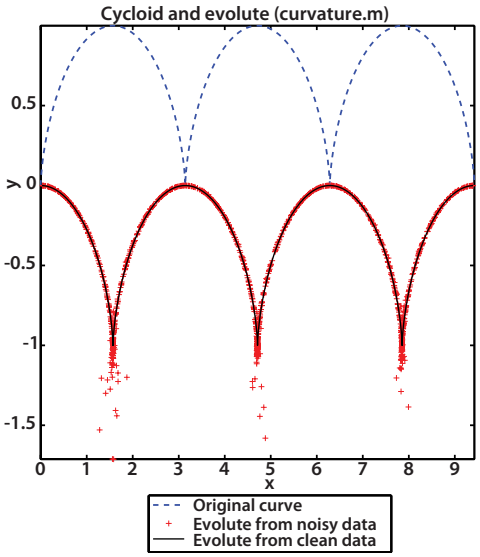


Figure 3.9: Cycloid and evolutes computed with code `Curvature.m`. The dashed blue line is the original cycloid; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

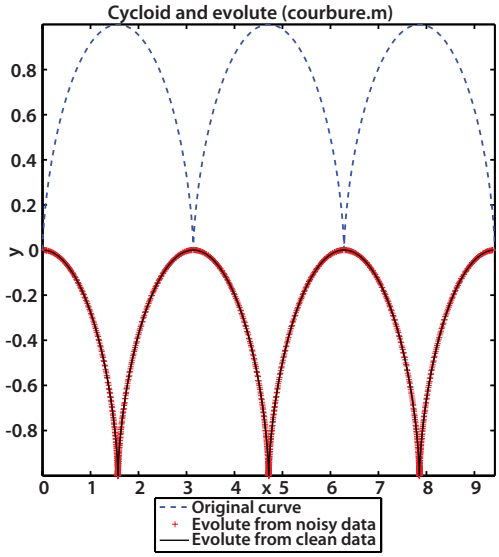


Figure 3.10: Cycloid and evolutes computed with code `Courbure.m`. The dashed blue line is the original cycloid; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

The evolute of a cycloid is another congruent cycloid translated downwards. Results of functions `Courbure.m` and `Curvature.m` are given in Figures 3.9 and 3.10 respectively.

3.3.1.6 Parabola

$$\begin{cases} x = 2 \cdot t \\ y = t^2 \end{cases}, \; t \in [-5;5] \tag{3.3.6}$$

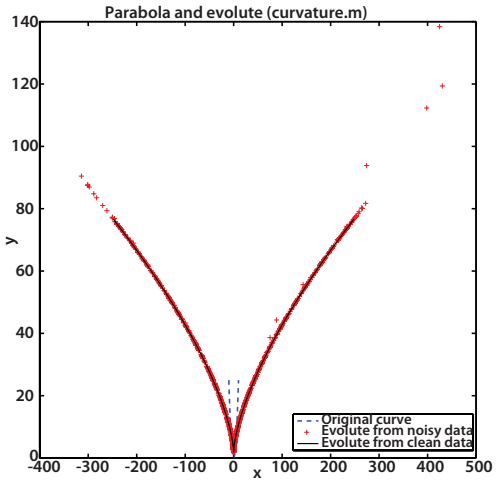


Figure 3.11: Parabola and evolutes computed with code `Curvature.m`. The dashed blue line is the original parabola; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

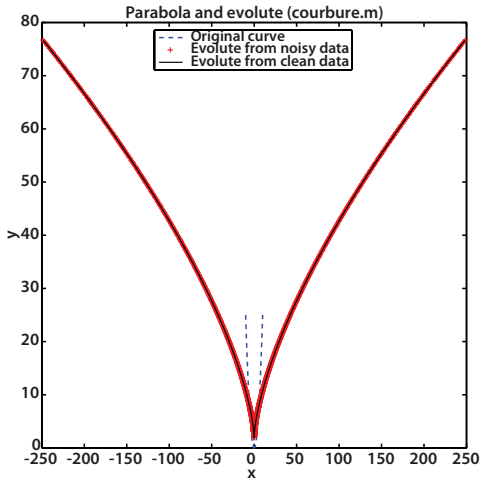


Figure 3.12: Parabola and evolutes computed with code `Courbure.m`. The dashed blue line is the original parabola; The continuous black line is its evolute computed from clean data, while the red crosses are the centers of curvature computed from noisy data.

The evolute of a parabola is a semicubical parabola (curve of shape $y = a \cdot |x|^{\frac{3}{2}}$). Results of functions `Courbure.m` and `Curvature.m` are given in Figures 3.11 and 3.12 respectively.

The evolutes of these curves are all known. In each case, the angle/time parameters θ or t contained 1000 points, and two versions of this parameter were

created: a "clean" one with regularly spaced points and a noisy one where Gaussian noise is added (to a maximum noise of half the space between θ, t points), allowing our testing of how the finite difference algorithms handles digitization scatter. X and Y variables are subsequently created for both the clean and noisy angle/time parameters and used as input of the `curvature.m` and `courbure.m` functions. As shown in plots 3.1 to 3.12, the `curvature.m` function handles all clean cases very well, but the noise creates problems at each cusp point of the evolutes, and makes the evolute of the logarithmic spiral much less accurate. Conversely, the `courbure.m` function handles all clean cases well except for the logarithmic spiral, whose centers of curvature get scattered far off their theoretical positions. However, this function handles noise better than the other one, as it does not increase that noise at cusp points of the evolute.

3.3.2 Theoretical example

Another test of the basic curvature codes (i.e. without preprocessing) and the full curvature codes (i.e. with smoothing) was done using the Von Schelling equation given in the introduction. We provide example plots of such curves for various values of the angle ω in figure 3.13.

3.3.2.1 Creation of numerical datasets

We have attempted to assess the performance of the code under conditions connected with human digitization of channel forms. For this purpose, we created two sets of data, one with a set of river shapes given by Von Schelling's equation,

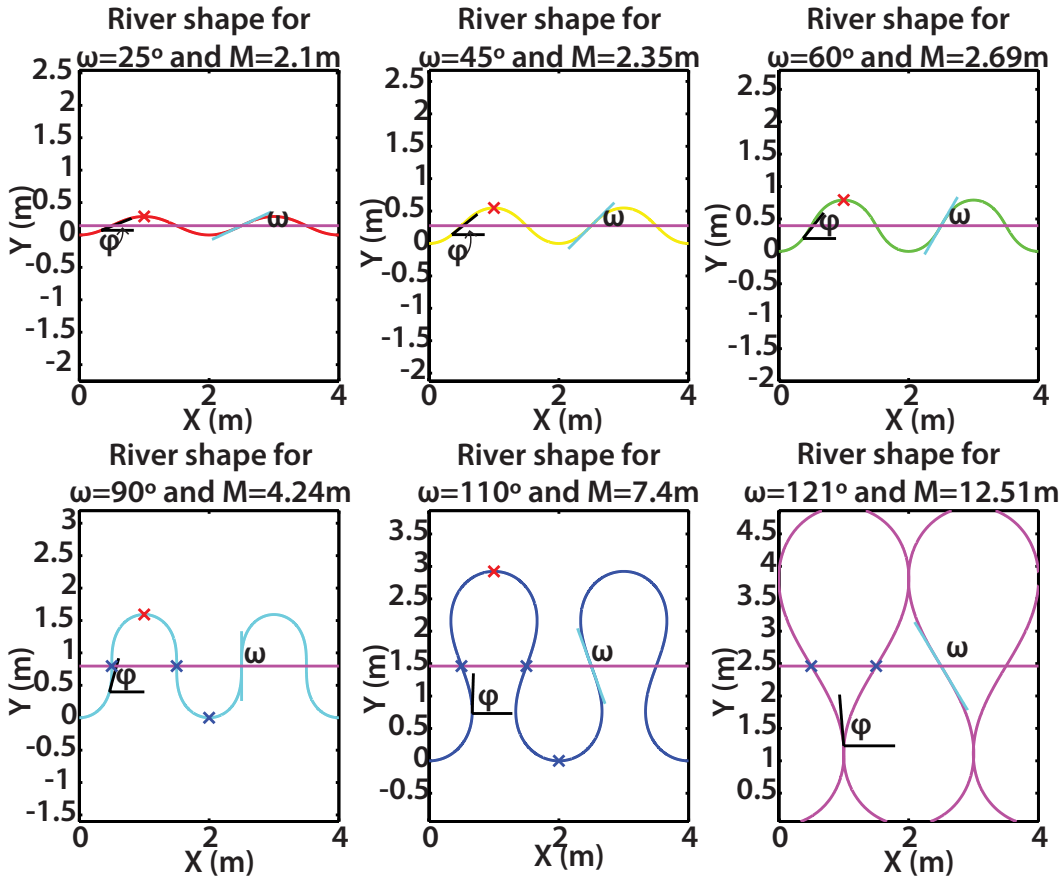


Figure 3.13: River shapes for six different values of ω. In each case, M and x were respectively chosen to yield a wavelength of 2 m and two spatial periods. The blue and red crosses are visual matches between the computed values of wavelength and amplitude and theoretical values given by equation (C.3.10) in C.3.

the other where noise was added to these shapes, and compared the unsigned radius of curvature values we obtained when running the code to the theoretical values at the same point. The datasets were designed as follows:

- "Clean data": We used a 402 values of ω ranging from 10^{-2} to 120.927 (maximum value at which two consecutive bends touch, numerical solution from equation (C.3.15) with $B = 0$ in C.3) and 16 values of M ranging from 0.01 to $3 \cdot \pi$. The abscissæ $X_{\omega,M}$ and ordinates $Y_{\omega,M}$ of each of those 6432 curves were created by summation of $\mathrm{d}X_{\omega,M}$ and $\mathrm{d}Y_{\omega,M}$ as given by equations (3.3.7) and (3.3.8); To make two periods (meanders) with spatial steps $\mathrm{d}x_M$ as small as possible and each curve as close as possible to its definition, vectors x_M (streamwise coordinates) included in each case $10,000$ points ranging from $x_M = 0$ to $x_M = 2 \cdot M$.

$$\begin{cases} \mathrm{d}X_{\omega,M}[i] = (x_M[i+1] - x_M[i]) \cdot \cos[\phi(x_M[i])] \\ \mathrm{d}Y_{\omega,M}[i] = (x_M[i+1] - x_M[i]) \cdot \sin[\phi(x_M[i])] \\ \phi_{\omega,M}(x[i]) = \omega \cdot \sin\left(\dfrac{2 \cdot \pi \cdot x[i]}{M}\right) \end{cases} \tag{3.3.7}$$

$$\begin{cases} X_{\omega,M}[i] = \displaystyle\sum_{j=1}^{i-1} \mathrm{d}X_{\omega,M}[j],\ X_{\omega,M}[0] = 0 \\ Y_{\omega,M}[i] = \displaystyle\sum_{j=1}^{i-1} \mathrm{d}Y_{\omega,M}[j],\ Y_{\omega,M}[0] = 0 \end{cases} \tag{3.3.8}$$

- "Noisy data": We used the basic curvature codes to compute the centers of curvature $O_{\omega,M}[i]$ (with coordinates $(OX_{\omega,M}[i], OY_{\omega,M}[i])$) of the clean curves $(X_{\omega,M}[i], Y_{\omega,M}[i])$ and extract the coordinates of the normal vectors $\overrightarrow{N_{\omega,M}}[i]$ for each point of each curve (equation (3.3.9)).

$$\begin{aligned}\overrightarrow{N_{\omega,M}}[i] &= \frac{\overrightarrow{N_{\omega,M_0}}[i]}{\left\|\overrightarrow{N_{\omega,M_0}}[i]\right\|} \\ \overrightarrow{N_{\omega,M_0}}[i] &= \begin{bmatrix} OX_{\omega,M}[i] - X_{\omega,M}[i] \\ OY_{\omega,M}[i] - Y_{\omega,M}[i] \end{bmatrix}\end{aligned} \tag{3.3.9}$$

We also computed, for a given curve, the "diagonal footprint" $\mathcal{D}_\lambda(M,\omega)$ of a meander, which we define as the diagonal of a rectangle whose sides are equal to a wavelength $\lambda(M,\omega)$ and to the amplitude $a(M,\omega)$ of a meander; $\mathcal{D}_\lambda(M,\omega)$ is computed using equation (3.3.10), and the formulæ for $\lambda(M,\omega)$ and $a(M,\omega)$ come from equation (C.3.10) in C.3.

$$\mathcal{D}_\lambda(M,\omega) = \sqrt{\lambda(M,\omega)^2 + a(M,\omega)^2} = M \cdot \sqrt{\mathbf{J}_0^2(\omega) + \frac{\mathbf{H}_0^2(\omega)}{4}} \tag{3.3.10}$$

For low ω angles, the diagonal footprint is much larger than the amplitude of the curve, and therefore much larger than the width of any river following that curve. Therefore, we define a "modified footprint" $\mathcal{D}'_\lambda(M,\omega)$, taken perpendicularly from the diagonal footprint. We plotted $\frac{\mathcal{D}_\lambda(M,\omega)}{M}$ and $\frac{\mathcal{D}'_\lambda(M,\omega)}{M}$ as functions of ω on figure 3.14.

$$\mathcal{D}'_\lambda(M,\omega) = \sqrt{[\sin(\omega)\cdot\lambda(M,\omega)]^2 + [\cos(\omega)\cdot a(M,\omega)]^2} \tag{3.3.11}$$

$$= M \cdot \sqrt{\sin^2(\omega)\cdot\mathbf{J}_0^2(\omega) + \cos^2(\omega)\cdot\frac{\mathbf{H}_0^2(\omega)}{4}} \tag{3.3.12}$$

Using Matlab©'s `randn` function and appropriate scaling, we created random, normalized Gaussian noise $\zeta_{\omega,M}[i]$ between -1 and $+1$, which was

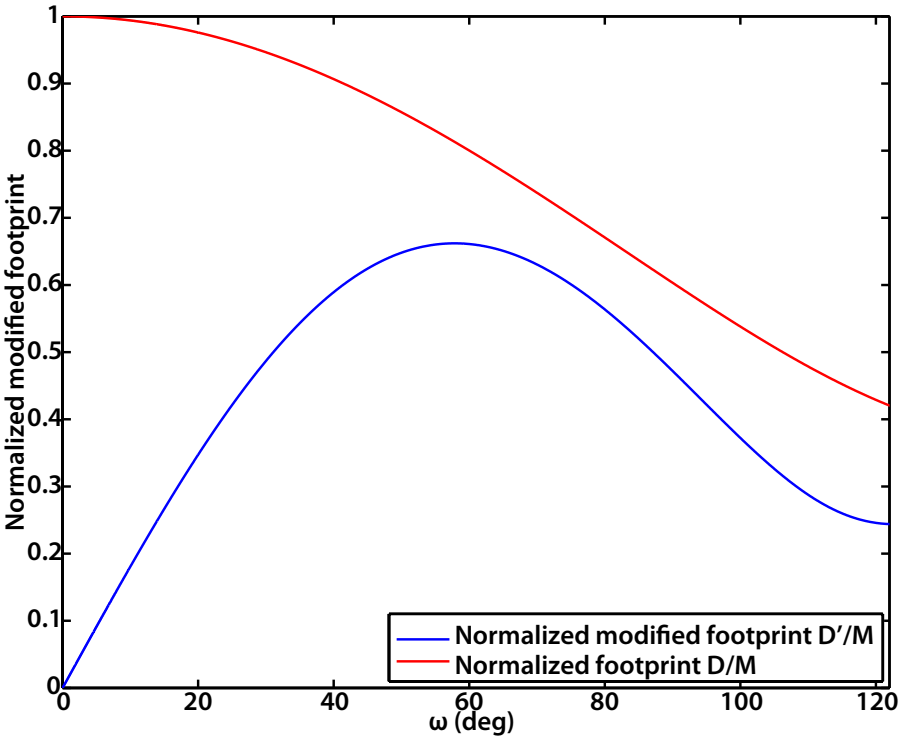


Figure 3.14: Normalized diagonal footprint (red, monotonically decreasing line) and normalized modified footprint (blue line) as functions of ω

then multiplied by $\frac{\mathcal{D}'_\lambda(M,\omega)}{100}$. This scaled noise was then multiplied by each normal vector and added to the coordinates $X_{\omega,M}[i]$ and $Y_{\omega,M}[i]$ of the clean curves, thus creating random normal noise around the clean curves of maximum amplitude $\pm\frac{\mathcal{D}'_\lambda(M,\omega)}{100}$ for a (ω,M) couple. Equation (3.3.13) defines the coordinates of these noisy curves, whose abscissæ we call $\chi_{\omega,M}[i]$ and whose ordinates we call $\xi_{\omega,M}[i]$.

$$\left.\begin{aligned}\chi_{\omega_k,M_j}[i] &= X_{\omega_k,M_j}[i] + \frac{\mathcal{D}'_\lambda\left(M_j,\omega_k\right)\cdot\zeta_{\omega_k,M_j}[i]}{1000}\cdot\overrightarrow{N_{\omega_k,M}}[i]\\ \xi_{\omega_k,M_j}[i] &= Y_{\omega_k,M_j}[i] + \frac{\mathcal{D}'_\lambda\left(M_j,\omega_k\right)\cdot\zeta_{\omega_k,M_j}[i]}{1000}\cdot\overrightarrow{N_{\omega_k,M_j}}[i]\end{aligned}\right\} \qquad (3.3.13)$$
$$\forall\,(k,j,i)\in[\![1;402]\!]\times[\![1;16]\!]\times[\![1;10,000]\!]$$

To simulate human, manual digitization (picking up points on a map with some radial error and an irregular step), we additionally reduced the sampling from $10,000$ to 781 points by randomly choosing integers $\{m_1,m_2,\ldots,m_p,\ldots\}$ from a uniform distribution between 1 and 25 using Matlab©'s `randi` function and then keeping only values of the index i separated by the random integers in the order they were drawn: $i_1=1$, $i_2=1+m_1$, $i_3=1+m_1+m_2$, $\ldots$, $i_{p+1}=1+\sum_{q=1}^{p}m_q$, $\ldots$. Enough integers were drawn to cover the full curves, and point $i=10,000$ was added as last point of each set when the last drawing would have produced an index i larger than $10,000$.

$$\left.\begin{aligned}\chi^{\text{sample}}_{\omega_i,M_j} &= \left\{\chi_{\omega_i,M_j}[1],\ldots,\chi_{\omega_i,M_j}\left[1+\sum_{q=1}^{p_k} m_q\right],\ldots,\chi_{\omega_i,M_j}\left[1+\sum_{q=1}^{p_{max}} m_q\right],10000\right\}\\ \xi^{\text{sample}}_{\omega_i,M_j} &= \left\{\xi_{\omega_i,M_j}[1],\ldots,\xi_{\omega_i,M_j}\left[1+\sum_{q=1}^{p_k} m_q\right],\ldots,\xi_{\omega_i,M_j}\left[1+\sum_{q=1}^{p_{max}} m_q\right],10000\right\}\end{aligned}\right\}$$
$$\forall (i,j,k) \in [\![1;402]\!]\times[\![1;16]\!]\times[\![1;739]\!] \tag{3.3.14}$$

3.3.2.2 Analysis applied to the numerical datasets

At high radii of curvature (curve locally close to a line, or neighborhoods of inflexion points where curvature changes signs), a small difference in curve coordinates will create a large difference in the value of the radius of curvature. The process of determining the radius of curvature is in that regard similar to determining the center of a circle with compass and ruler only. In that process, one would draw two chords and find the intersection of their perpendicular bisectors; however, two chords subparallel to each other would yield a much lower accuracy for determining the center than two chords close to perpendicularity. Therefore, we derived the equation for the radius curvature as a function of curvilinear distance – equation (C.2.7) of C.2 –. For every value of ω and M, we compared the value of the radius of curvature obtained through our codes with the theoretical value; the basic curvature codes, `curvature.m` and `courbure.m` were used on the clean dataset to assess their respective accuracies, while the full `width.m` code was used in "curvature-only" mode to assess its accuracy on noisy data. A full Chebyshev window of 81 points was used to filter the noisy data.

The maximum error recorded for `curvature.m` on the clean data is nearly

identical for all values of ω and M and is close to 1%; the average error is 1.15‰ and the minimum error is comparable to a computer's accuracy (in the 10^{-9}%). While the second run with a 45^o-rotated dataset may take care of some problems caused by the finite-differencing algorithm when using `courbure.m`, it also fails on others, creating *some* inaccurate values of the radius of curvature for $\omega \geq \frac{\pi}{2}$, usually in no more than 2 or 3 points. These problems are very limited and occur at vertical tangents oriented (90^o-orientation) more rarely at tangents oriented at 45^o, as the second run makes them vertical. A difference of 10% between computation and theory is found in 6% of the calculation. Outside of these few localized areas, the accuracy of `courbure.m` has the same magnitude as that of `curvature.m` on clean data. To get a representative view of the accuracy of the code, we chose to display graphically the 10^{th}, 50^{th} and 90^{th} percentiles of the error, as a function of ω, all values of M being superimposed on the same graph. These plots were made for the `width.m` code using both `curvature.m` and `courbure.m`. Additionally, two types of error have been analyzed: the error e between signed radii of curvature and the error $|e|$ between unsigned radii of curvature. We display such plots in figures 3.15 and 3.16.

Finding a good evaluation tool for the accuracy of the code on noisy data is more difficult, as noise still propagates somehow to the radius of curvature values, and some narrow peeks of high inaccuracy prevent statistical properties such as minimum, maximum, or average from being representative of the general accuracy of the code. Superimposed plots of the computed radius of curvature versus the theoretical one for all values of ω and M would be illegible. Therefore, we chose to

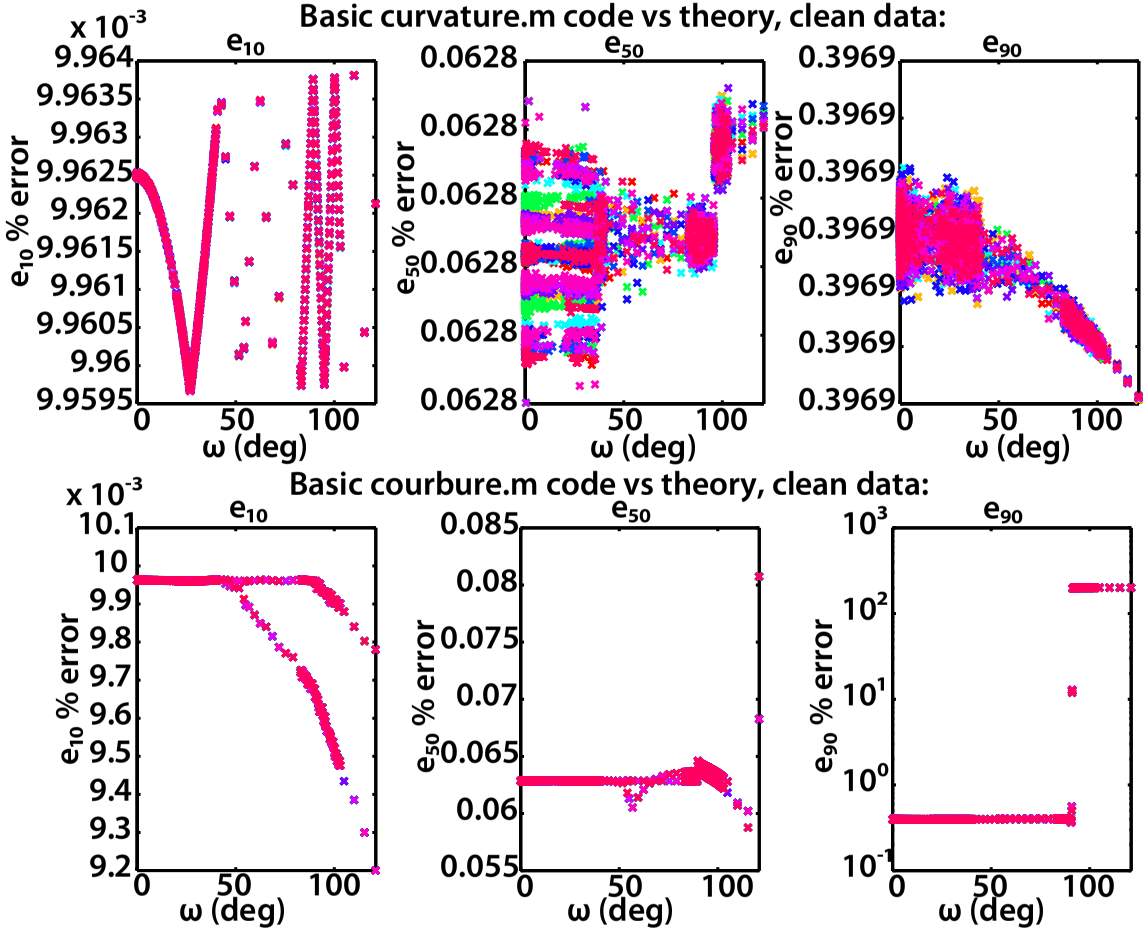


Figure 3.15: 10^{th}, 50^{th} and 90^{th} percentiles of the error between computed and theoretical radius of curvature, as functions of ω, for both `curvature.m` and `courbure.m` basic codes.

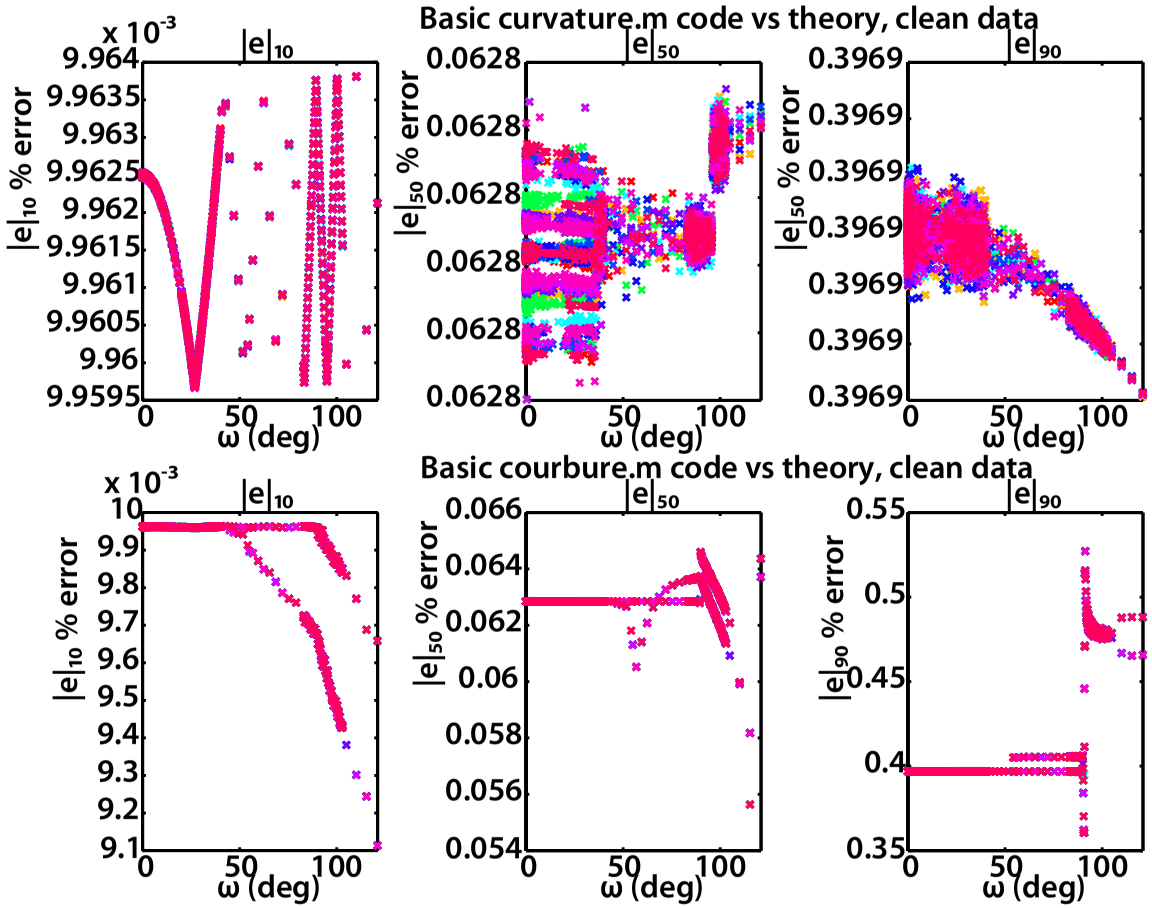


Figure 3.16: 10^{th}, 50^{th} and 90^{th} percentiles of the error between absolute values of computed and theoretical radius of curvature, as functions of ω, for both `curvature.m` and `courbure.m` basic codes.

only display the error percentile plots in figures 3.17 and 3.18.

As is the case with the basic curvature codes, the error of the full `width.m` code using `curvature.m` or `courbure.m` as basic functions are similar, except for $\omega \leq \frac{\pi}{2}$ angles, where `courbure.m` may get highly inaccurate, as the very high e_{90} values show; $|e|_{90}$ also increases sharply after 90^o for `courbure.m` but less than e_{90}, which indicates that some of the inaccuracy due to `courbure.m` comes from a difference in signs of the computed radii of curvature. Both codes are more accurate around $\omega \simeq \frac{\pi}{6}$, while small angles ($\omega < 10^o$) and larger angles show more inaccuracies. For all values of ω and M, e_{10} and $|e|_{10}$ stay between 0.5% and 3.5%, e_{50} and $|e|_{50}$ stay between 5% and 13%. e_{90} ranges from 20% and 140% for `curvature.m`, from 20% to 230% for `courbure.m` while both `curvature.m` and `courbure.m` show $|e|_{90}$ ranging from 20% to 110%.

3.3.3 Mississippi river

We applied the code in "Curvature-only" mode to analyze the relationship between geometrical parameters of the channel (bottom angle, total relief — depth —, width, bottom length) and the local radius of curvature. We used to this effect a dataset from the US Army Corps of Engineers (1974 Mississippi River Hydrographic Survey) that contains transects located by their 3D-coordinates. This dataset was converted to a set of 2D transects defining the cross-sectional form of the channel and one 2D planview defining the sinuous form of the channel; the area of study in provided in Figure 3.19. Planform resolution is good with 90% of all studied centerline points separated by no more than 367 m, while cross-sectional

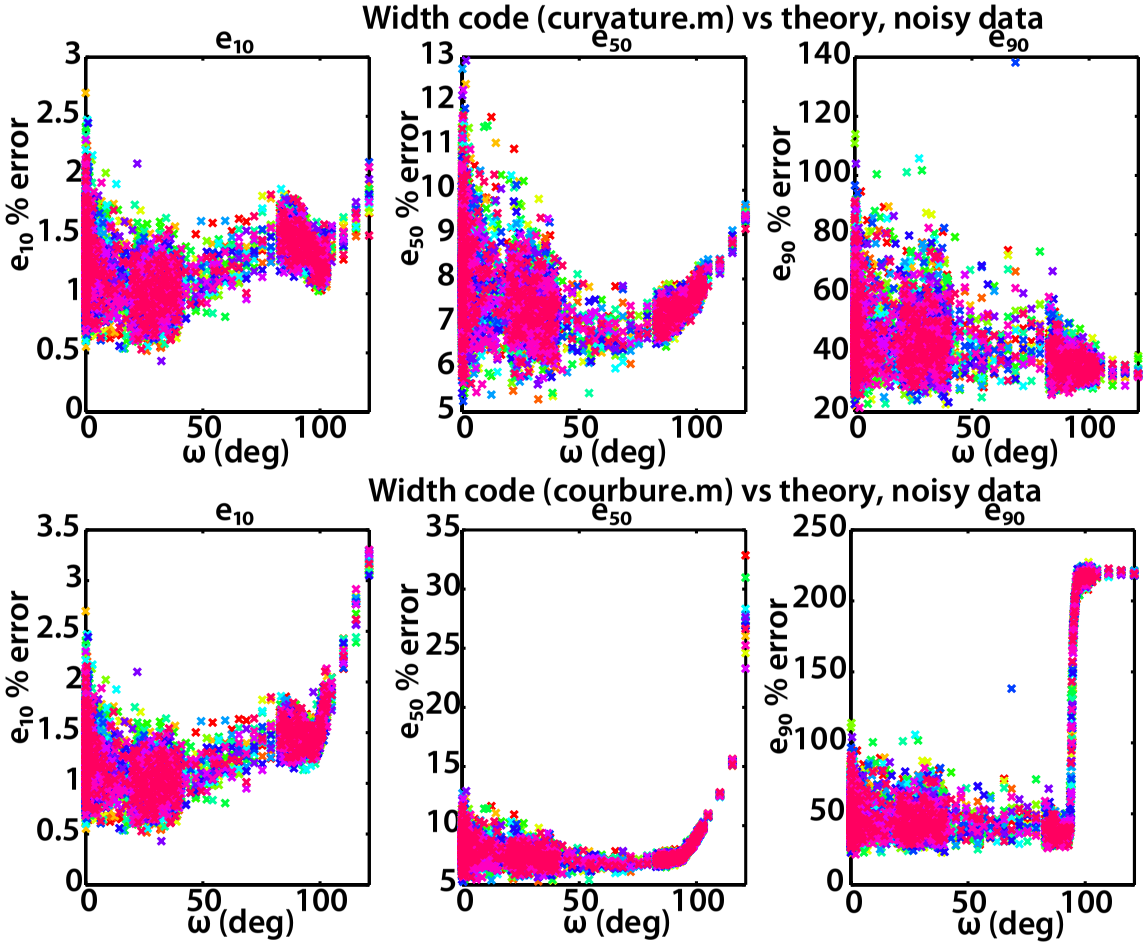


Figure 3.17: 10^{th}, 50^{th} and 90^{th} percentiles of the error between computed and theoretical radius of curvature, as functions of ω, for full `width.m` code using `curvature.m` and `courbure.m` basic codes.

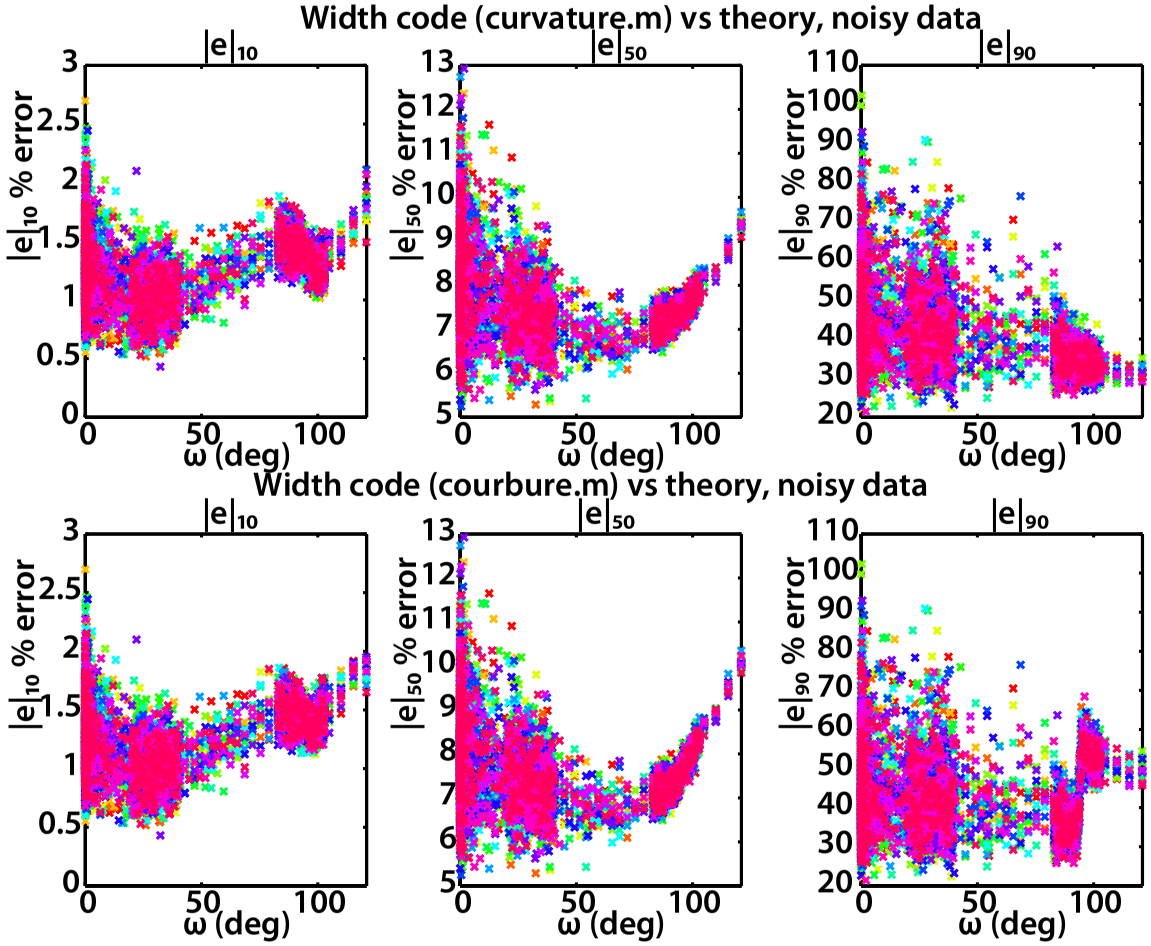
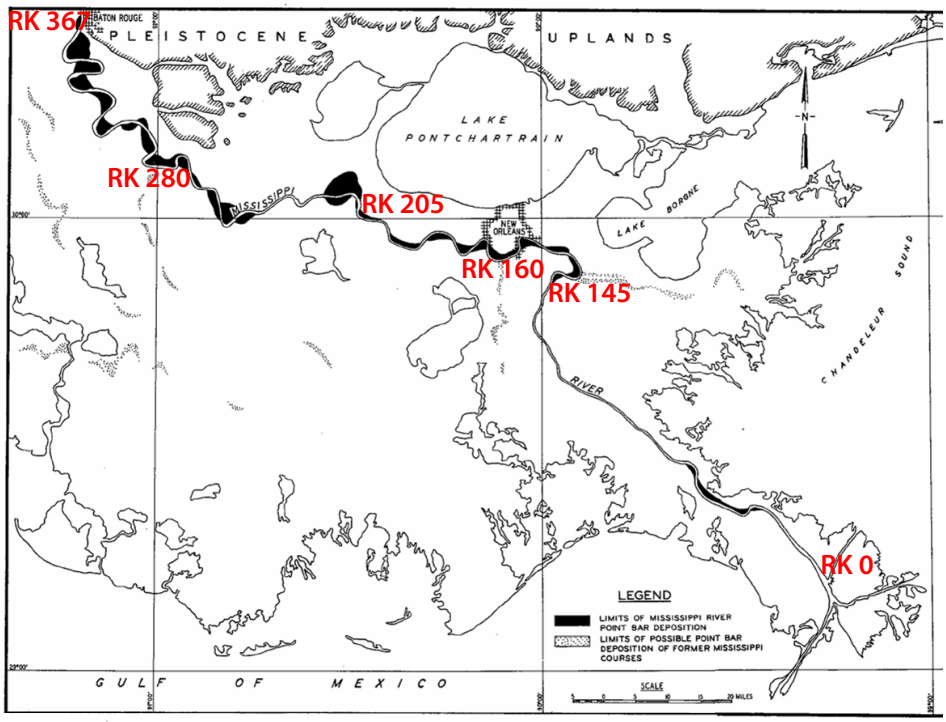


Figure 3.18: 10^{th}, 50^{th} and 90^{th} percentiles of the error between absolute values of computed and theoretical radius of curvature, as functions of ω, for full `width.m` code using `curvature.m` and `courbure.m` basic codes.



Source: US Army Engineer Waterways Experiment Station (USAEWES)

Figure 3.19: Mississippi river and area of study.

resolution varies from poor to good with a median horizontal distance between transect points equal to 24.5 m but some distances between transect points larger than a kilometer. We used for this analysis $N = 1,030$ transects from the lowermost part of the Mississippi River, corresponding to the first 300 riverkm upstream from Head of Passes (HoP) and manually fit a quadrangle to each of the transects, which we used to compute cross-sectional parameters; we computed the river radius of curvature from the planview centerline. We provide six of these vertical, cross-stream transects with our manually-fit quadrangles as examples in figure 3.20. In addition, we used data from Hudson and Kesel [2000] to get migration rates between 1877 and 1924.

These sides of a quadrangle were chosen to represent the left sidewall (side 1 of length $d_1[i]$, from point $P_1[i]$ to point $P_2[i]$), the bottom of the channel (side 2 of length $d_2[i]$, from $P_2[i]$ to $P_3[i]$), the right sidewall (side 3 of length $d_3[i]$, from $P_3[i]$ to $P_4[i]$) and the river surface or width (side 4 of length $d_{\text{top}}[i]$, from $P_4[i]$ to $P_1[i]$). In a given transect, coordinates of points $P_j[i]$, $(i,j) \in [\![1;N]\!] \times [\![1;4]\!]$, are called $u_j[i]$ (abscissæ) and $v_j[i]$ (ordinates). Given that the river centerline coordinates are filtered before being processed by the curvature code (yielding values of the radius of curvature $R[i]$), we also filter the transect data. However, the river width may be variable, and the transect misaligned, and therefore we first normalize and align the transects before smoothing their coordinates. We define the normalized coordinates

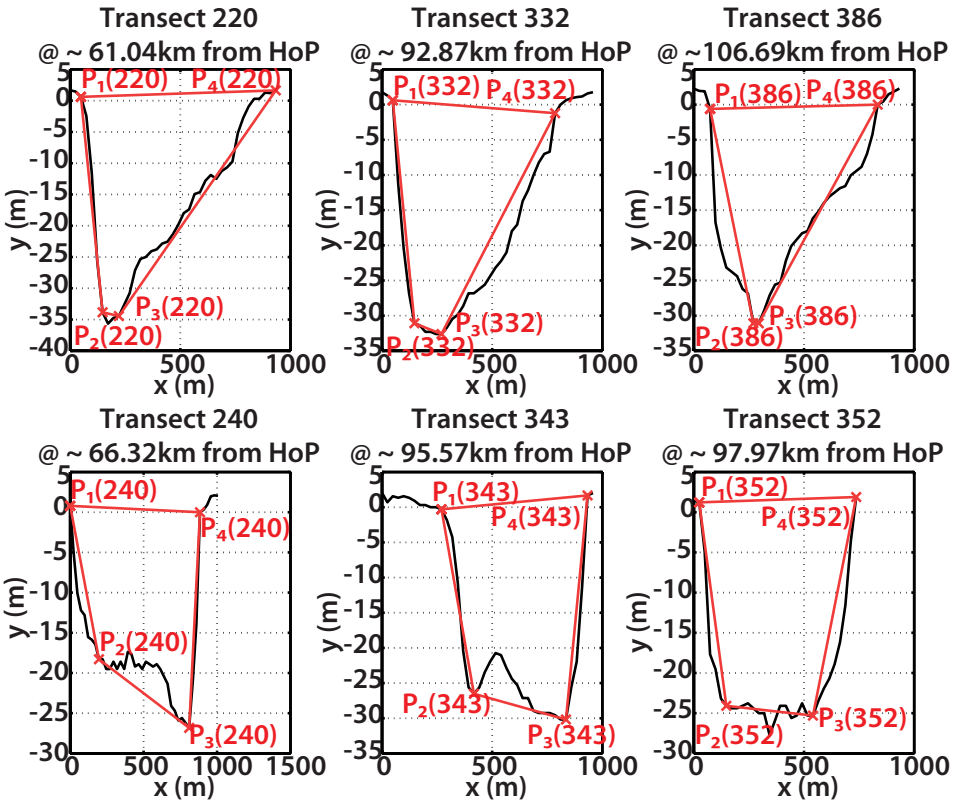


Figure 3.20: Six transects from the Lower Mississippi (black continuous lines) and corresponding fitted quadrangles (four red segments). River kilometers upstream of Head of Passes (HoP) indicated at the top of each transect may be located using Figure 3.19. Transects 240 and 343 show the presence of bedforms on the river bottom.

with equation (3.3.15):

$$\begin{cases} u_j^n[i] = \dfrac{1,000}{B[i]} \cdot \left(u_j[i] - \min\limits_j u_j[i] \right) \\ v_j^n[i] = \dfrac{1,000}{B[i]} \cdot \left(v_j[i] - \min\limits_j v_j[i] \right) \end{cases} \tag{3.3.15}$$

where B is the channel width defined as

$$B = \begin{cases} u_4 - u_2 - (u_1 - u_2) \cdot \dfrac{v_4 - v_2}{v_1 - v_2} & \text{if } v_1 > v_4 \\ u_3 - u_1 + (u_4 - u_3) \cdot \dfrac{v_1 - v_3}{v_4 - v_3} & \text{if } v_1 < v_4 \end{cases} \tag{3.3.16}$$

These coordinates $\left(u_j^n\right)$ are such that the leftmost point has a 0 abscissa, the highest point has a 0 ordinate, and $\left\| \overrightarrow{P_1[i]P_4[i]} \right\| = 1,000 \ \forall i \in [\![1;N]\!]$. These normalized coordinates are then filtered with the same process as for the radius of curvature but different windows. For the centerline, we use the parameters $L = 81$ (length of window) and $\theta = 0.6$, which combine an 81-point Chebyshev window with a 41-point Ricker window into a window $\mathcal{W}_C$; however, we only use a 41-point Ricker window $\mathcal{W}_T$ for the transect smoothing. We process the data with the Matlab© function `filtfilt` (which performs a zero-phase filtering) as described in equation (3.3.17):

$$\begin{cases} u_j^s = \texttt{filtfilt}(\mathcal{W}_T, 1, u_j^n) \\ v_j^s = \texttt{filtfilt}(\mathcal{W}_T, 1, v_j^n) \end{cases} \tag{3.3.17}$$

We are then able to compute the bottom slope S_2^s, the total relief H^s and the bottom length L^s

$$S_2^s[i] = \arctan\left(\frac{v_3^s[i]-v_2^s[i]}{u_3^s[i]-u_2^s[i]}\right) \quad (3.3.18)$$

$$H^s[i] = \max\left(v_1^s, v_4^s\right) - \min\left(v_2^s, v_3^s\right) \quad (3.3.19)$$

$$L^s[i] = \sqrt{\left(u_3^s-u_2^s\right)^2+\left(v_3^s-v_2^s\right)^2} \quad (3.3.20)$$

We introduce the variable $\mathcal{G}$, which can be any of these four geometrical parameters ($\left|S_2^s\right|$, H^s, B^s and L^s). Due to the phase shift between the channel shape in planform and the channel shape in cross-section, a direct match between $\mathcal{G}[i]$ and $R[i]$ is not possible. Therefore, we plot $\mathcal{G}$ and $|R|$ as functions of upstream distance on the same graph, and manually match the local extrema of these variables (for instance the minima of $\left|S_2^s\right|$ with the maxima of $|R|$ and vice-versa) using the method described in section C.1.3 of Appendix C.

We display plots showing $\mathcal{G}$ and $|R|$ as functions of upstream distance from Head of Passes in figures 3.21 to 3.28, where $\mathcal{G}$ is $\left|S_2^s\right|$ in figures 3.21 and 3.21, H^s in figures 3.23 and 3.24, B^s in figures 3.25 and 3.26 and L^s in figures 3.27 and 3.28.

Once the extrema of $\mathcal{G}$ and $|R|$ are found and numbered, we can manually match them in $(\mathcal{G}[i_m], |R[i_n]|)$ couples. To extend the correlations to intermediary points (points between the extrema), we match values of $\mathcal{G}$ and $|R|$ using point indices. That is, if the extrema $\mathcal{G}[k]$ and $\mathcal{G}[m]$ respectively correspond to the extrema $|R[l]|$and $|R[n]|$ (two consecutive mimima or two consecutive maxima), we

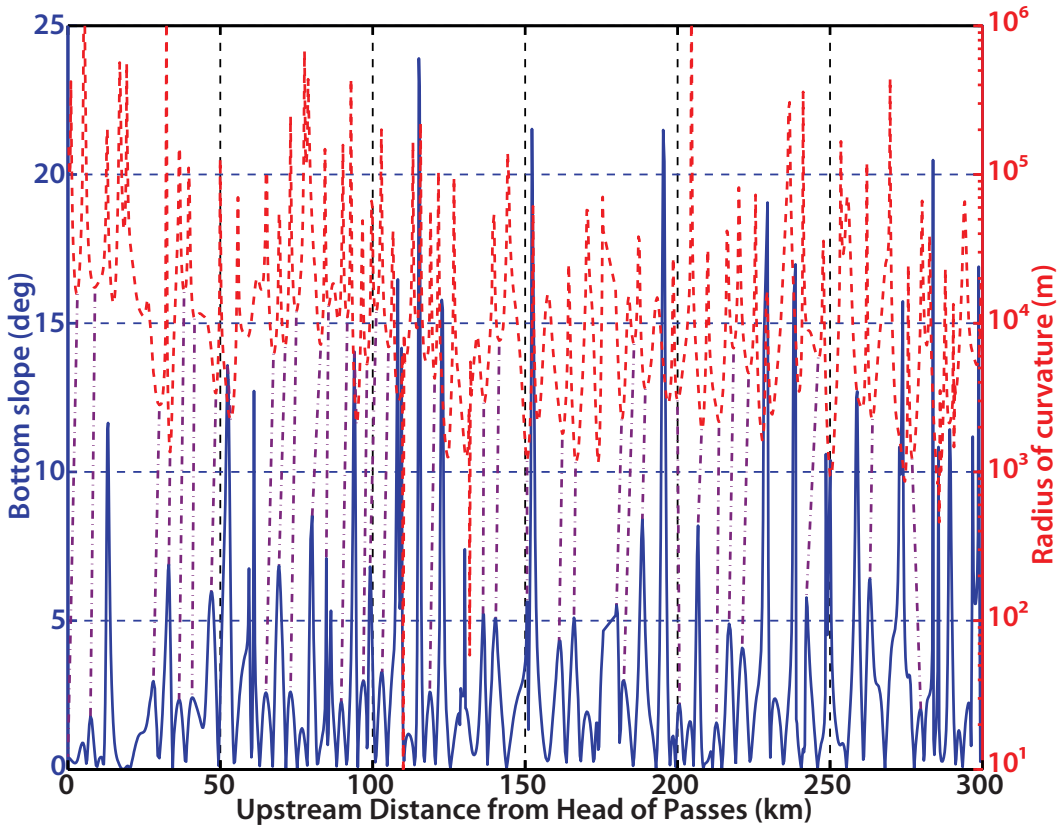


Figure 3.21: Bottom slope $\left|S_2^s\right|$ (continuous blue line, linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Minima of $|R|$ are linked to corresponding maxima of $\left|S_2^s\right|$ by purple dot-dashed lines.

first compare the number of points M and N of both variables between the extrema: $M = \mathrm{Card}_{i\in[\![k;m-1]\!]}\left(\mathcal{G}\left[i\right]\right) = m-k$, $N = \mathrm{Card}_{j\in[\![l;n-1]\!]}\left(\left|R\left[j\right]\right|\right) = n-l$. The smaller of these two numbers is used to pick-up points in the other variable. For instance, if $M < N$, then we associate $\mathcal{G}\left[k\right]$, $\mathcal{G}\left[k+1\right]$, $\mathcal{G}\left[k+2\right]$, …, $\mathcal{G}\left[m-1\right]$ to (respectively) $\left|R\left[l\right]\right|$, $\left|R\left[l+\lfloor\frac{N}{M}\rceil\right]\right|$,$\left|R\left[l+\lfloor 2\cdot\frac{N}{M}\rceil\right]\right|$, …, $\left|R\left[n-1\right]\right|$ (where $\lfloor x\rceil$ is the nearest-integer rounded value of x). Conversely, if $M > N$, we associate we asso-

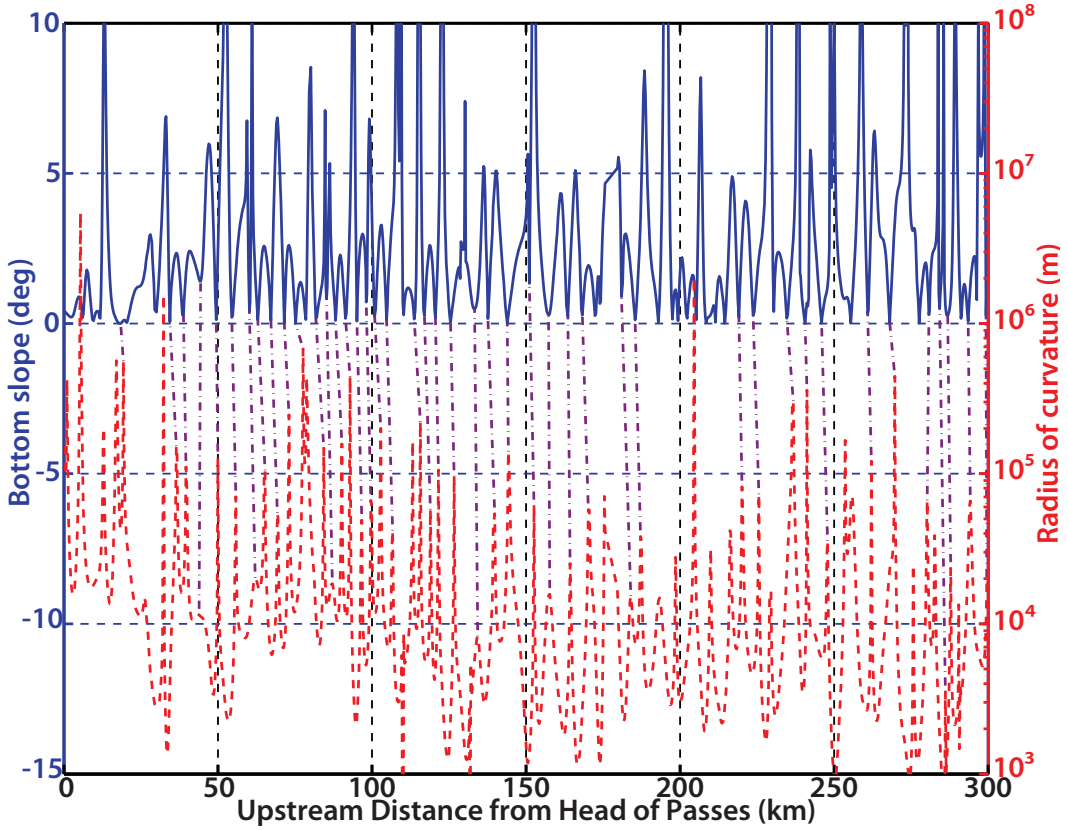


Figure 3.22: Bottom slope $\left|S_2^s\right|$ (continuous blue line, linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Maxima of $|R|$ are linked to corresponding minima of $\left|S_2^s\right|$ by purple dot-dashed lines.

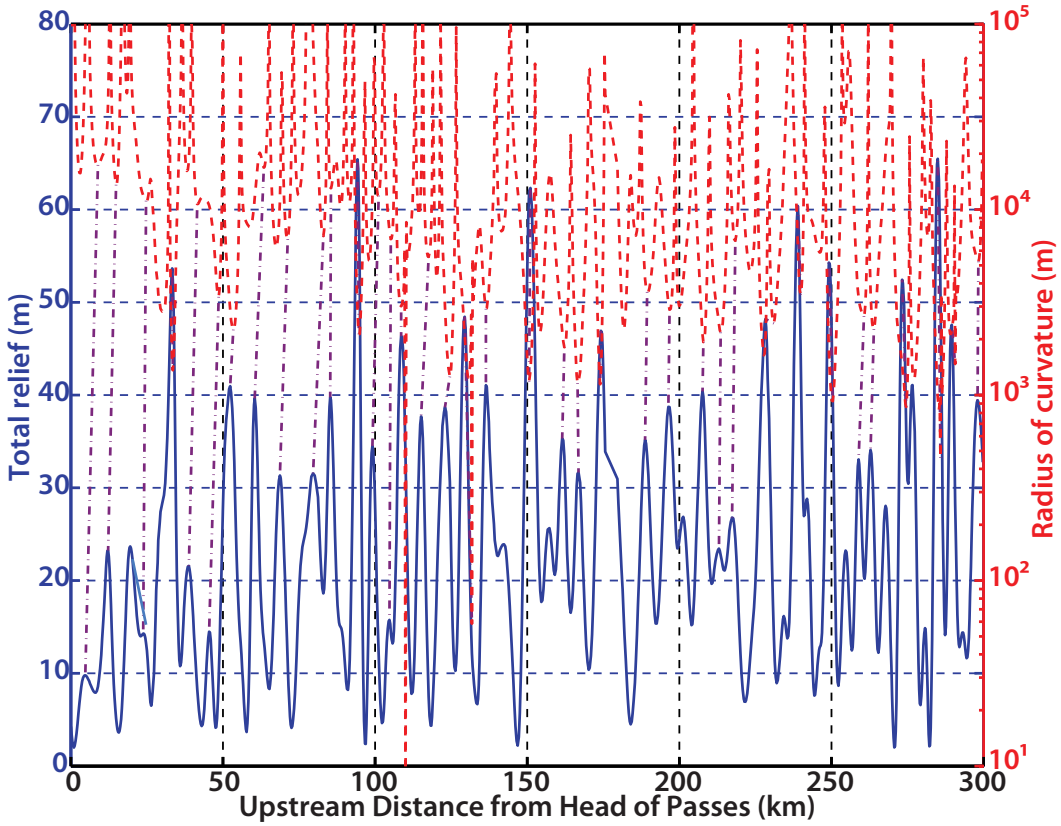


Figure 3.23: Total relief H^s (continuous blue line, linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Minima of $|R|$ are linked to corresponding maxima of H^s by purple dot-dashed lines.

ciate $|R[l]|$, $|R[l+1]|$, $|R[l+2]|$, …, $|R[n-1]|$ to (respectively) $\mathcal{G}[k]$, $\mathcal{G}\left[k+\lfloor\frac{M}{N}\rceil\right]$, $\mathcal{G}\left[k+\lfloor 2\cdot\frac{M}{N}\rceil\right]$, …, $\mathcal{G}[l-1]$.

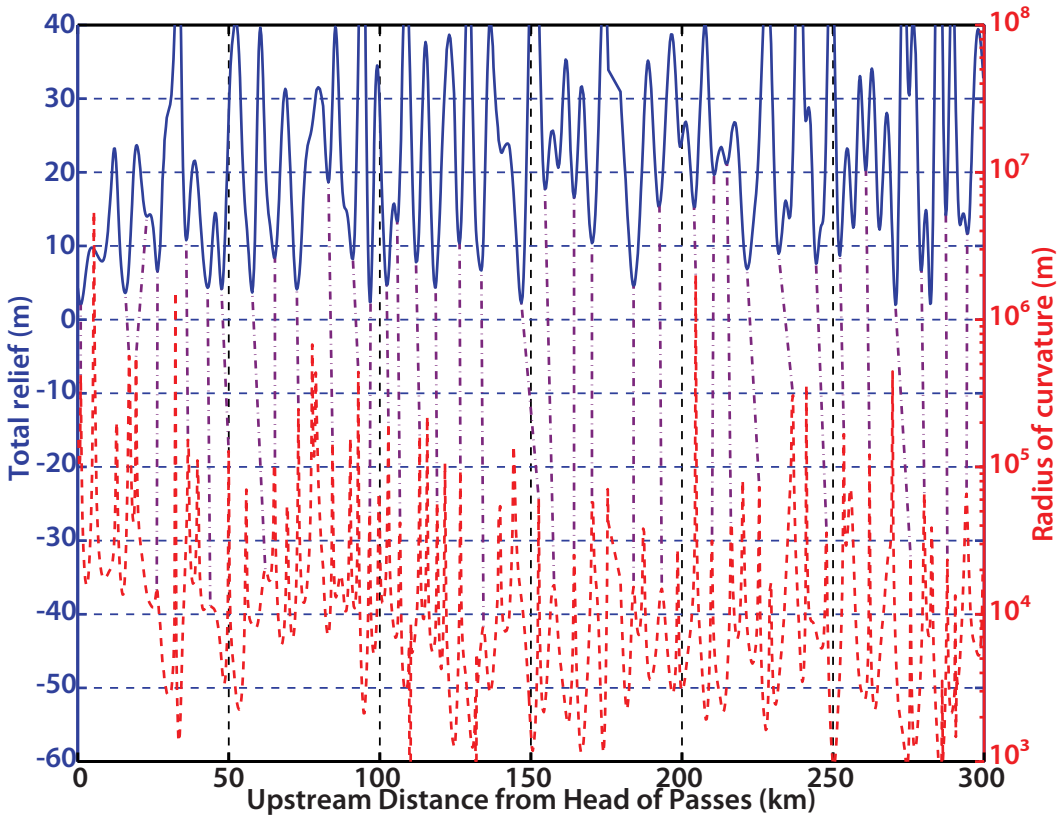


Figure 3.24: Total relief H^s (continuous blue line, linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Maxima of $|R|$ are linked to corresponding minima of H^s by purple dot-dashed lines.

We show resulting cross-plots and orthogonal least-square fits in figures 3.29 to 3.35. When two different types of correlation (semi-logarithmic or power-law) seemed possible, we chose the relationship after comparison of correlation coefficients: the higher correlation coefficient determined the chosen correlation type. The "correlation index" $\mathcal{I} = \frac{R_\perp}{N}$ that corresponds to a given least-square fit is displayed on each plot next to the equation of that fit. We define this index as the

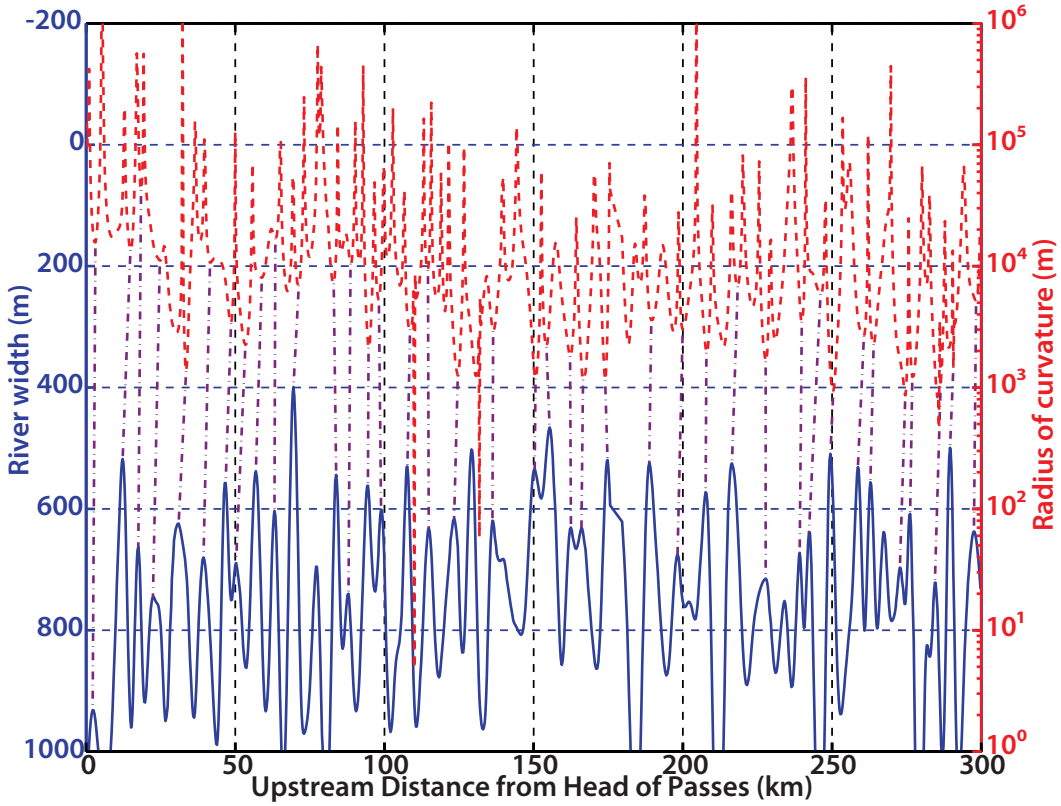


Figure 3.25: Channel width B^s (continuous blue line, reverse linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Minima of $|R|$ are linked to corresponding minima of B^s by purple dot-dashed lines.

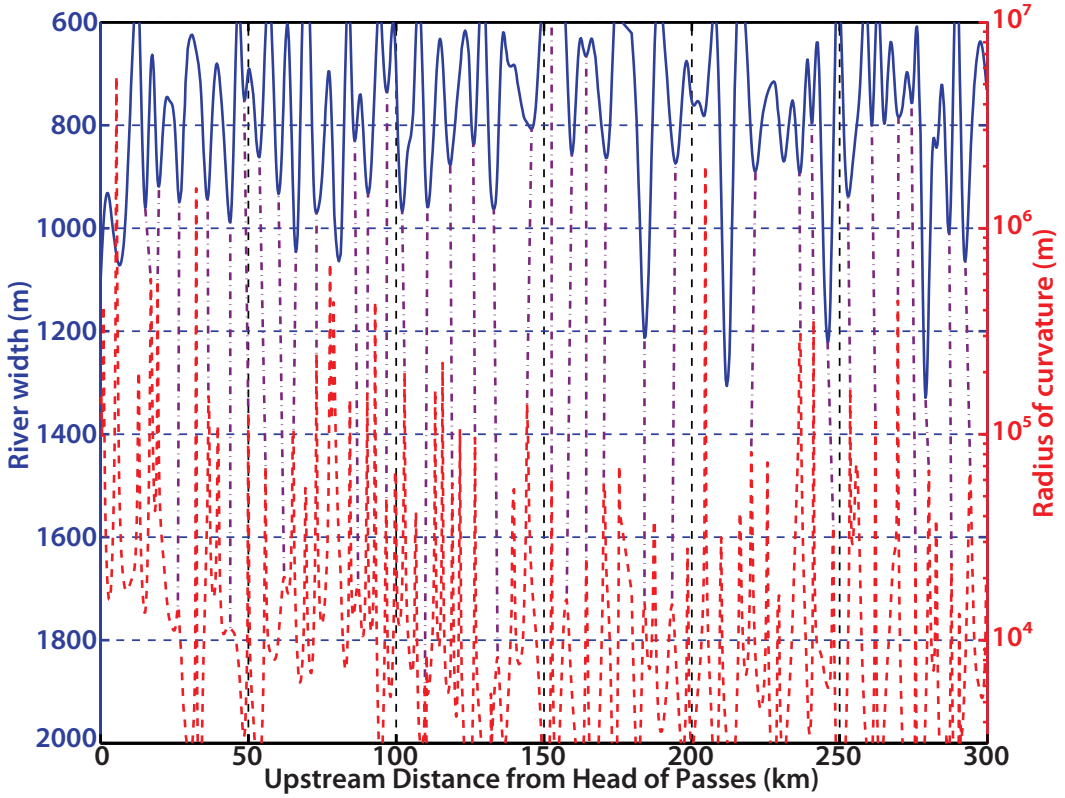


Figure 3.26: Channel width B^s (continuous blue line, reverse linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Maxima of $|R|$ are linked to corresponding maxima of B^s by purple dot-dashed lines.

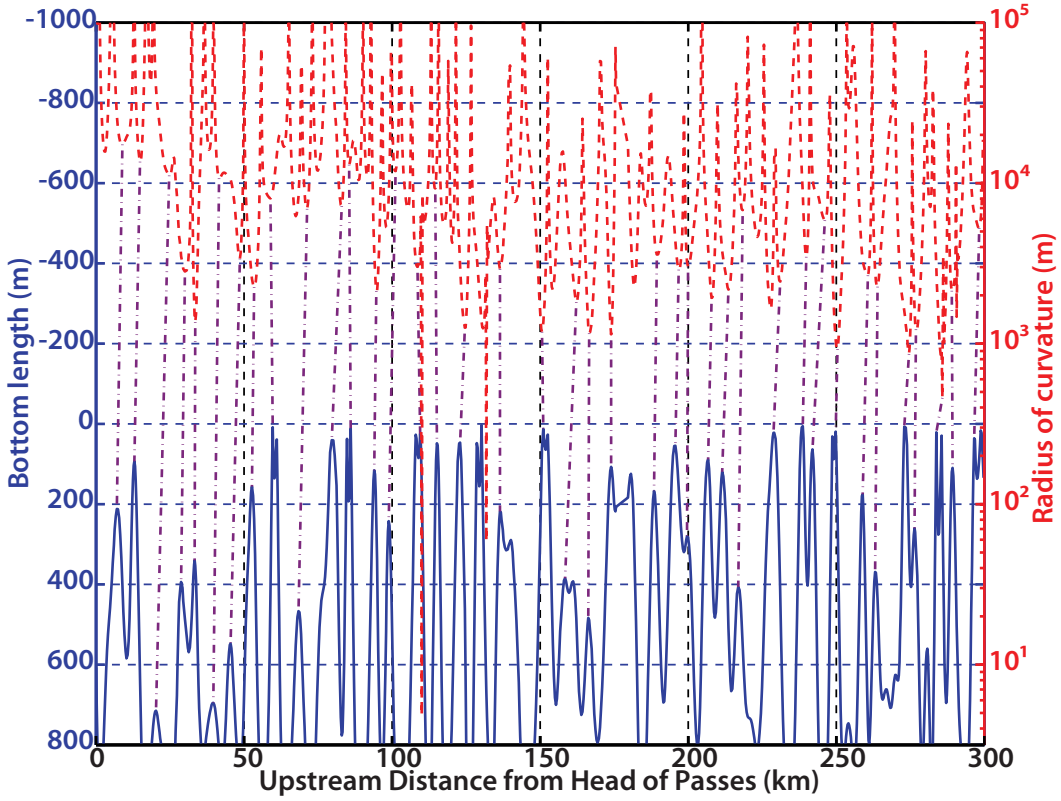


Figure 3.27: Bottom length L^s (continuous blue line, reverse linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Minima of $|R|$ are linked to corresponding minima of L^s by purple dot-dashed lines.

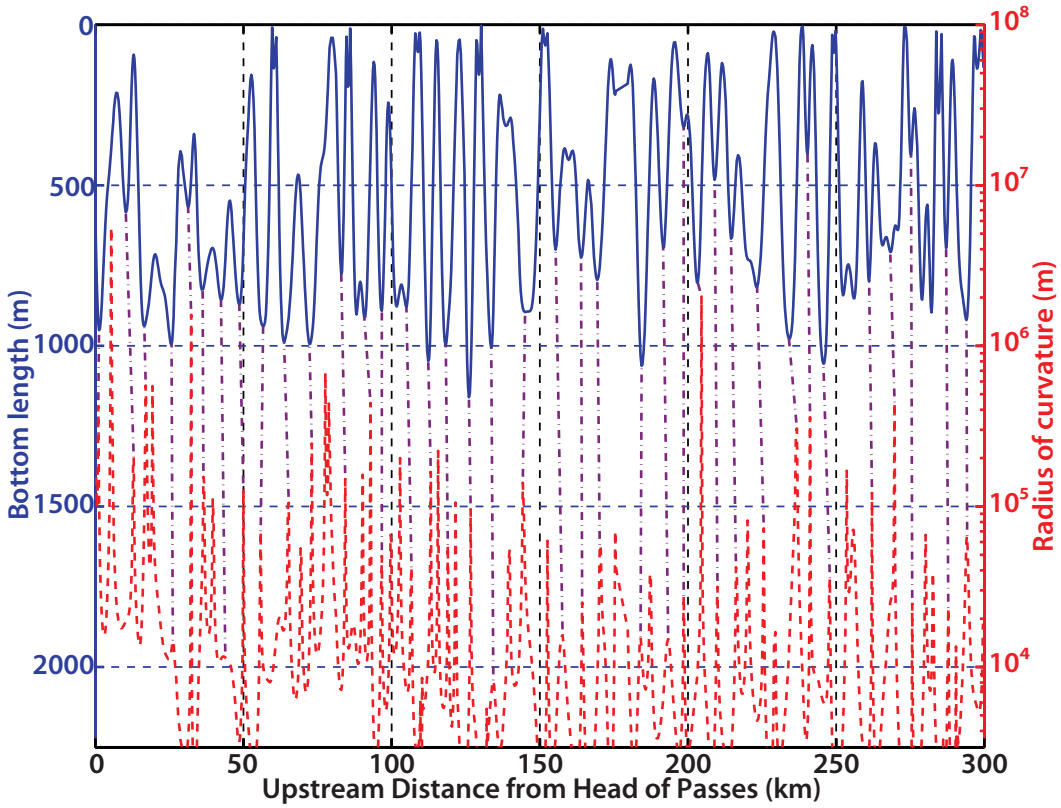


Figure 3.28: Bottom length L^s (continuous blue line, reverse linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. Maxima of $|R|$ are linked to corresponding maxima of L^s by purple dot-dashed lines.

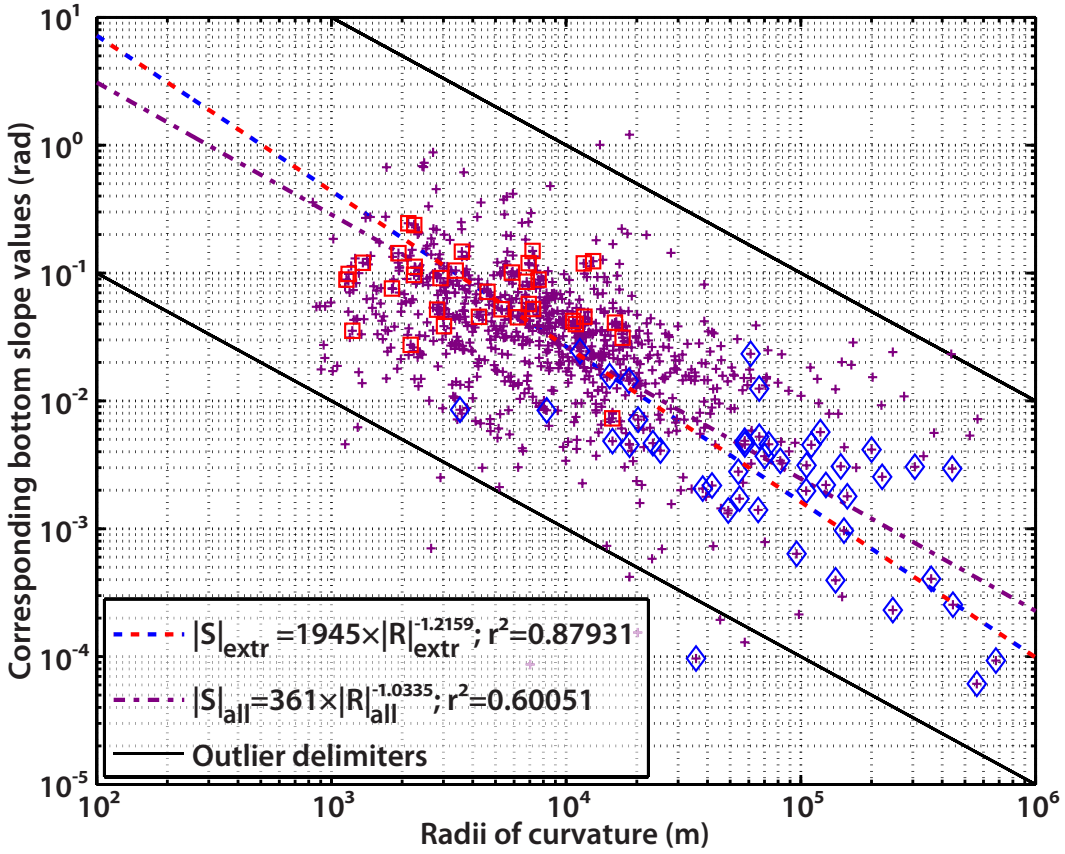


Figure 3.29: Bottom slope $\left|S_2^s\right|$ as function of corresponding values of unsigned radius of curvature $|R|$. Maxima of $\left|S_2^s\right|$ with corresponding minima of $|R|$ are red squares. Minima of $\left|S_2^s\right|$ with corresponding maxima of $|R|$ are blue diamonds. Intermediary values of $\left|S_2^s\right|$ and $|R|$ are dark purple crosses. The dashed red and blue line is the least-square fit for extrema only and takes into account all extrema, while the dot-dashed dark purple line is the least-square fit for all values of $\left|S_2^s\right|$ and $|R|$ and does not take the visually-assessed outliers into account. This figure shows that tighter bends of the lower Mississippi are associated with steeper slopes.

sum of the residuals $R_\perp = \sum_i d\,(X_i, \text{line})$ divided by the number of points N. Because each fit has a different slope, the residuals have different units and cannot be compared directly. Their values are, however, indicators of the goodness of the fit (for a fixed number of points, a larger spread means a larger $\mathbb{J}$ index).

Although scattering is significant, most probably because of the low sampling rate of each transect and the ambiguity of some transects, the cross-plots show clear trends: the bottom slope is almost exactly proportional to the river curvature ($|S \cdot R| \simeq$ constant), the total relief decreases with a power of the radius of curvature ($|H| \simeq \frac{H_0}{R^\varepsilon}$, $\varepsilon \simeq 0.36$), the river width increases slightly with the radius of curvature ($|B| \simeq B_0 \cdot |R|^\eta$, $\eta \simeq 0.06$) and the bottom length also increases with the radius of curvature ($L \simeq L_0 + L_1 \cdot \log_{10}(|R|)$). These trends show that the Mississippi River is shallow and has a flat and long bottom in linear parts of meanders, and is deep with a very short bottom in the bends. A cross-section of the Mississippi river will therefore resemble a deep triangle in tight meanders and a trapezoid in linear areas of the River, which explains the results found by [Nittrouer et al., 2011] concerning the presence of sediment cover in straight reaches but the absence of sediment cover in scoured, tighter bends (see Figures 3.32 and 3.33). The migration rate is low, with an average of 3.41 $\text{m} \cdot \text{yr}^{-1}$ [Hudson and Kesel, 2000] , which is equivalent to 0.44% of the median width ($B_{50} = 774$ m) per year. Additionally, the distributions of left- and right-bank slopes, which we plot in figure 3.36, are extremely similar.

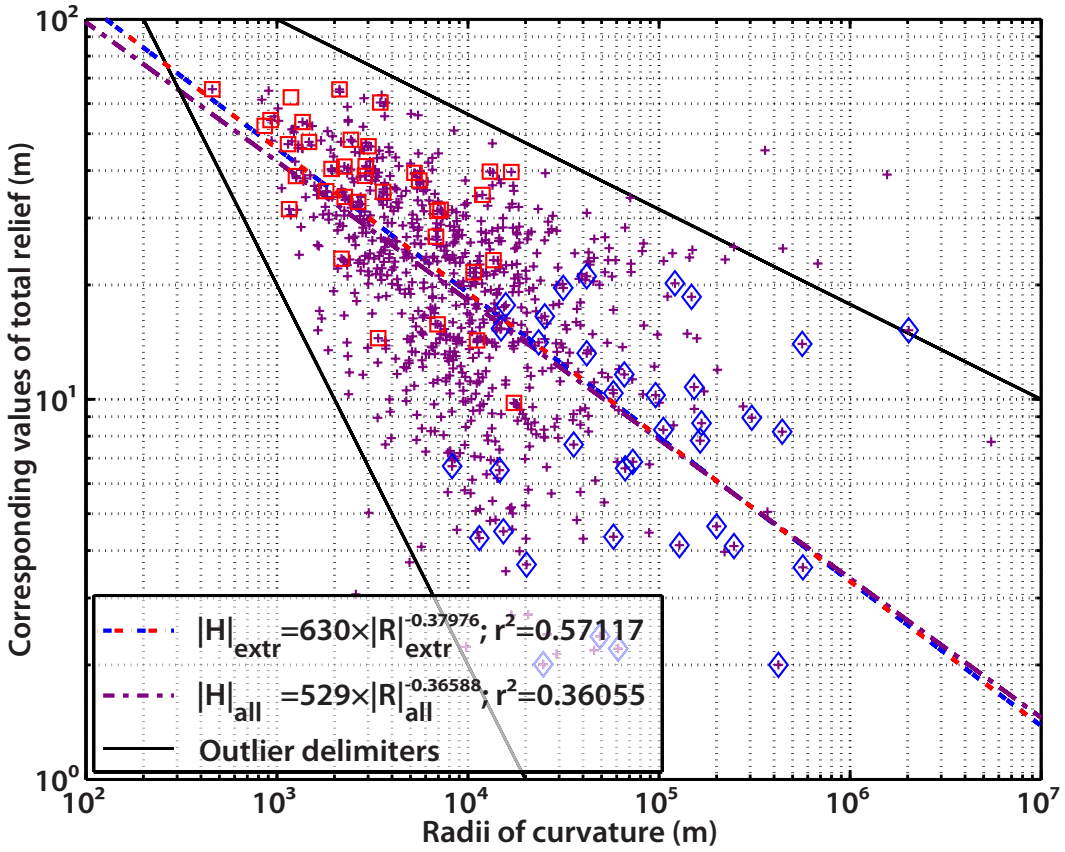


Figure 3.30: Total relief $|H^s|$ as function of corresponding values of unsigned radius of curvature $|R|$. Maxima of $|H^s|$ with corresponding minima of $|R|$ are red squares. Minima of H^s with corresponding maxima of $|R|$ are blue diamonds. Intermediary values of $|H^s|$ and $|R|$ are dark purple crosses. The dashed red and blue line is the least-square fit for extrema only and takes into account all extrema, while the dot-dashed dark purple line is the least-square fit for all values of $|H^s|$ and $|R|$ and does not take the visually-assessed outliers into account. This figure shows that tighter bends of the lower Mississippi are associated with a deeper channel.

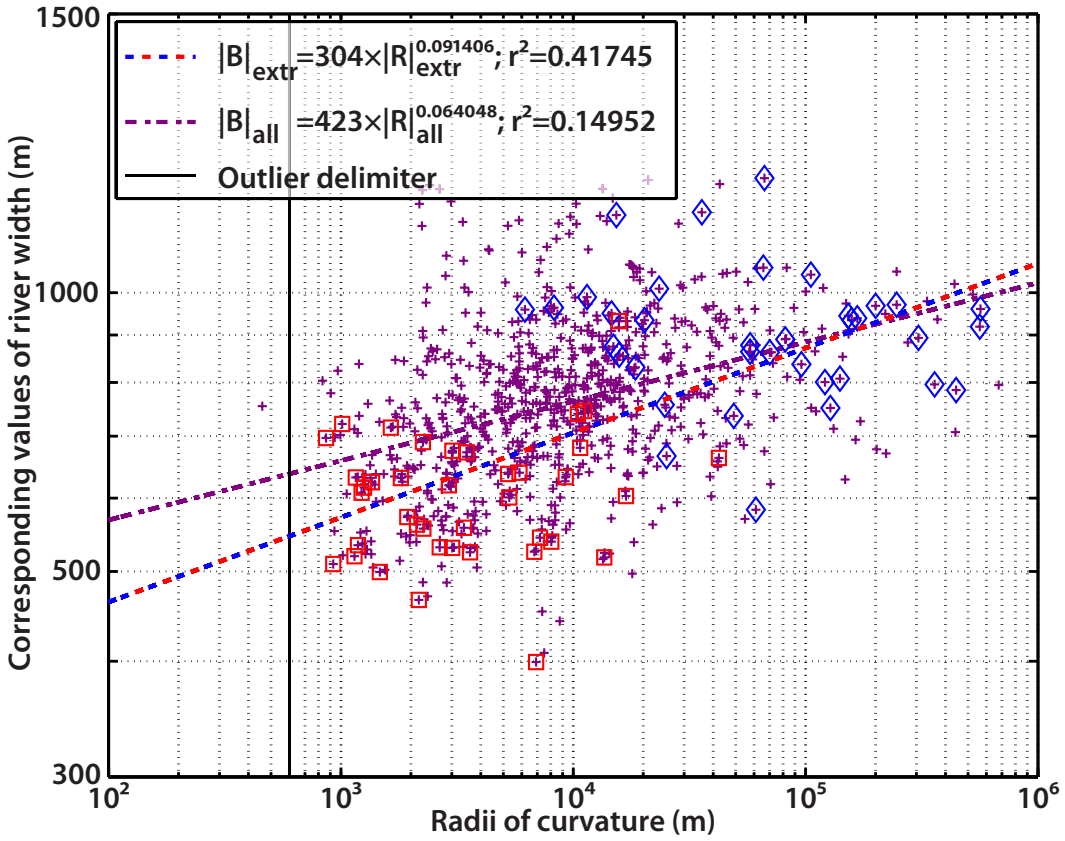


Figure 3.31: Channel width B^s as function of corresponding values of unsigned radius of curvature $|R|$. Minima of B^s with corresponding minima of $|R|$ are red squares. Maxima of B^s with corresponding maxima of $|R|$ are blue diamonds. Intermediary values of B^s and $|R|$ are dark purple crosses. The dashed red and blue line is the least-square fit for extrema only and takes into account all extrema, while the dot-dashed dark purple line is the least-square fit for all values of B^s and $|R|$ and does not take the visually-assessed outliers into account. This figure shows that tighter bends of the lower Mississippi are associated with a narrower channel.

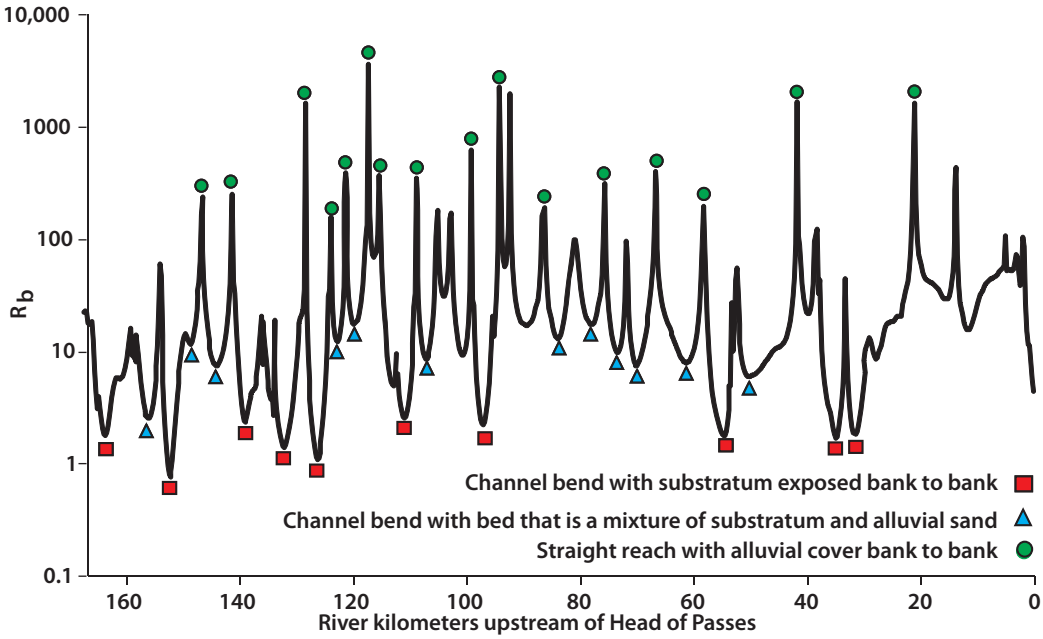


Figure 3.32: Radius of curvature divided by median channel width (R_b) for the lower 165 km of the Mississippi River (referenced to distance above the outlet at Head of Passes). Tight bends containing "channel-bottom substratum" and lacking alluvial cover are labelled with red squares; subtle bends with a mixture of "channel-bottom substratum" and "alluvial-sediment facies" are labelled with blue triangles; straight-reach segments with sandy alluvium bank to bank are labelled with green circles. From [Nittrouer et al., 2011].

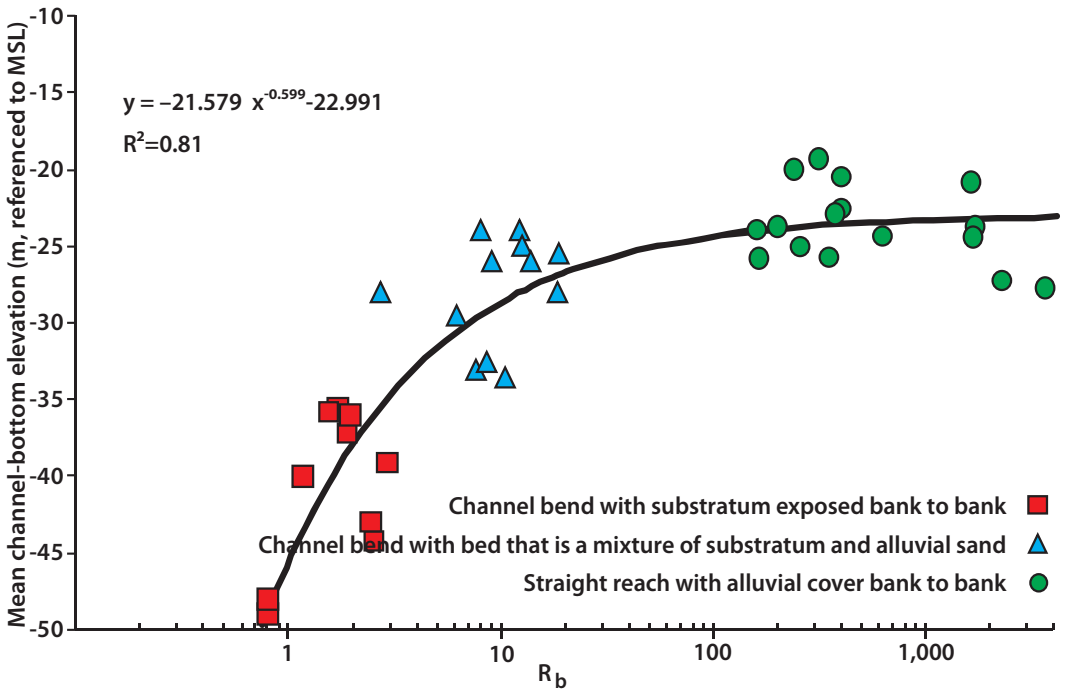


Figure 3.33: Mean channel-bed elevation and radius of curvature divided by median channel width (R_b) for the reaches labelled in Figure 3.32. Tight bends containing "channel-bottom substratum" and lacking alluvial cover are labelled with red squares; subtle bends with a mixture of "channel-bottom substratum" and "alluvial-sediment facies" are labelled with blue triangles; straight-reach segments with sandy alluvium bank to bank are labelled with green circles. From [Nittrouer et al., 2011].

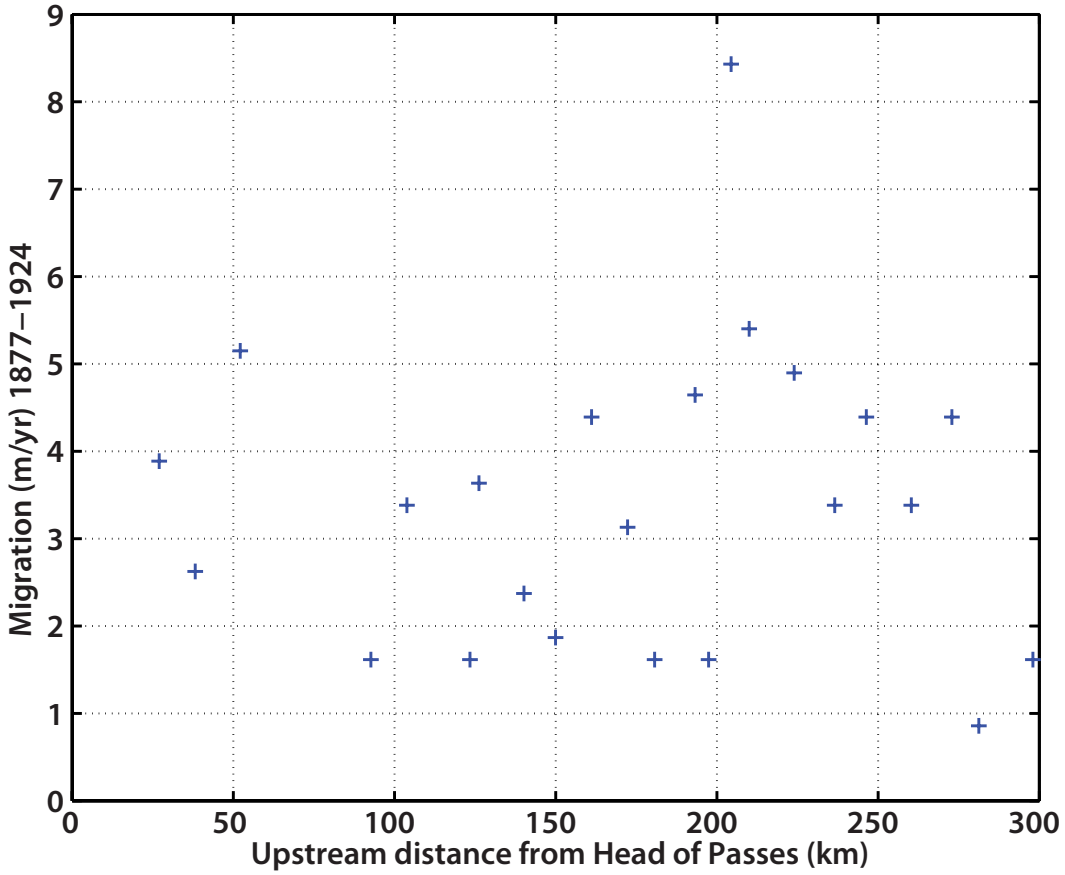


Figure 3.34: Migration rates of the Mississippi river from 1877 to 1924 as a function of upstream distance from Head of Passes. Data from Hudson and Kesel [2000].

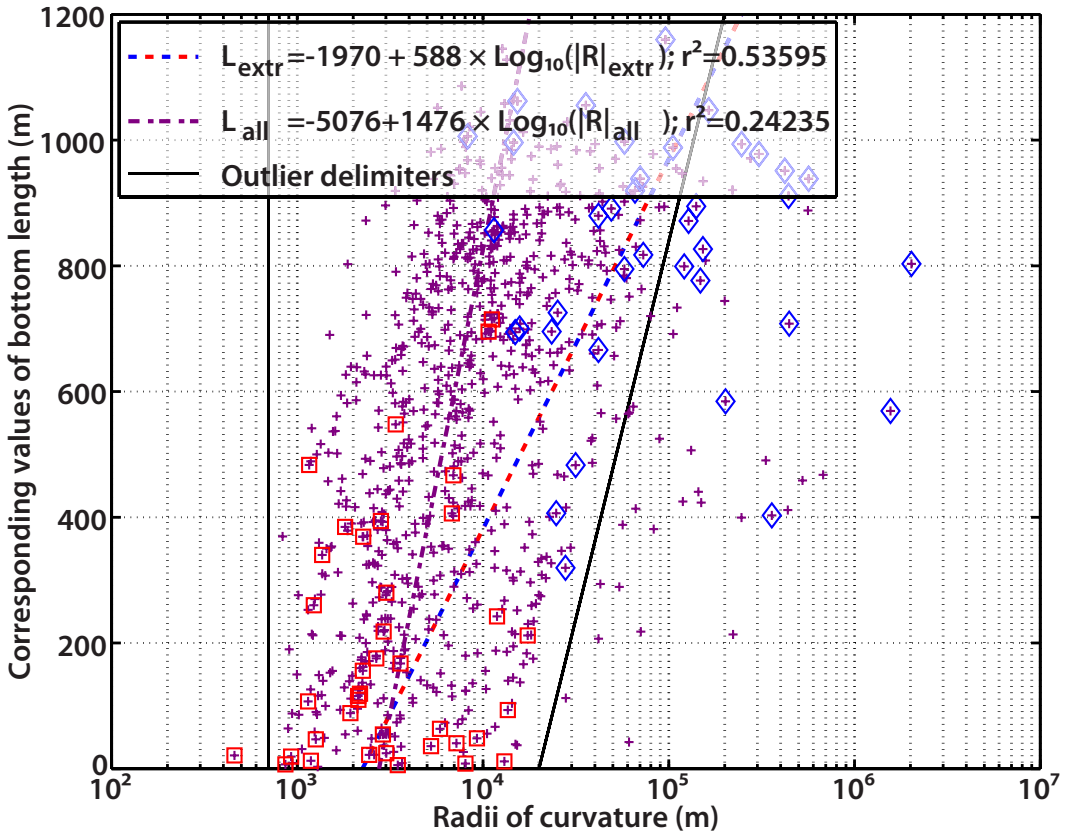


Figure 3.35: Bottom length L^s as function of corresponding values of unsigned radius of curvature $|R|$. Minima of L^s with corresponding minima of $|R|$ are red squares. Maxima of L^s with corresponding maxima of $|R|$ are blue diamonds. Intermediary values of L^s and $|R|$ are dark purple crosses. The dashed red and blue line is the least-square fit for extrema only and takes into account all extrema, while the dot-dashed dark purple line is the least-square fit for all values of L^s and $|R|$ and does not take the visually-assessed outliers into account. This figure shows that tighter bends of the lower Mississippi are associated with a shorter bottom length.

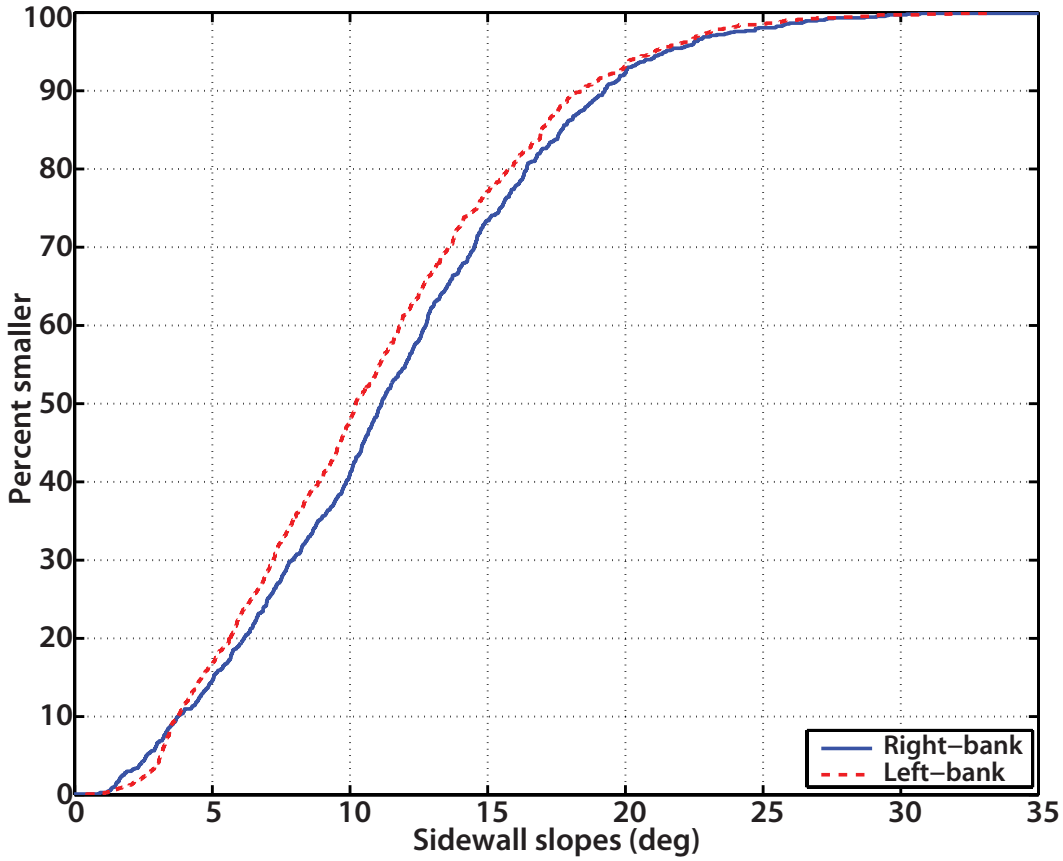


Figure 3.36: Cumulative distribution functions of left-bank (red dashed line) and right-bank (blue continuous line) sidewall slopes.

We also analyzed the delay (lag or advance) between extrema of the radius of curvature and extrema of the other geometry parameters. A cumulative distribution plot of this delay is provided in Figure 3.37.

The cumulative distributions plots show that even though there is a small difference in the delays between the geometry variables and curvature, the distribution trends are very similar. Additionally, the median delay is approximately 1 km for all variables; this median delay is equivalent with one channel width in the studied section of the river.

3.3.4 Trinity River

We also applied the full code to analyze the relationship between geometrical radius of curvature and width of the channel for the Trinity River in eastern Texas. The dataset was digitized from orthophotos (dated 2009) of the river, beginning at Lake Livingston Dam (LLD) and ending 180 km downstream of LLD, and consisted of centerline ($4,857$ points), left- ($5,973$ points) and right-outer banks ($6,108$ points). The resolution of this dataset is excellent, with the 90^{th} percentile for distance between centerline points being only 156 m. No channel cross-sections were available in this dataset. An additional set of orthophotos from 1996 was also digitized to evaluate the migration rate of the Trinity river. The area of study is outlined in Figure 3.38

We downsampled the centerline to $1,349$ points to avoid error due to oversampling; indeed, a very fine sampling of the data will make segments joining consecutive points too close to being parallel, which decreases the accuracy of the code

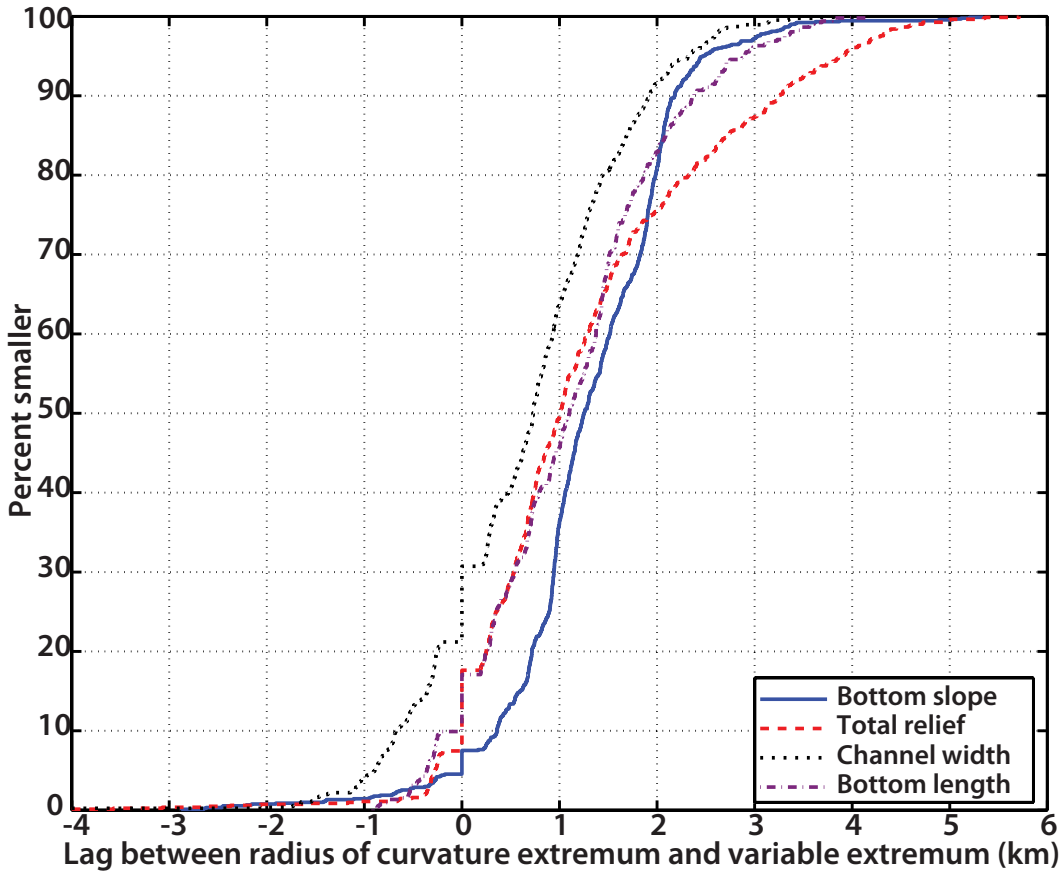


Figure 3.37: Delay between $|R|$ and the corresponding values of the geometric parameters of the Mississippi River. Respective delays are represented by a continuous blue line for the bottom slope (data assembled from Fig. 3.21 and 3.22), a dashed red line for the total relief (data assembled from Fig. 3.23 and 3.24), a dotted black line for the channel width (data assembled from Fig. 3.25 and 3.26) and a dot-dashed purple line for the bottom length (data assembled from Fig. 3.27 and 3.28). We define the delay as the distance to an extremum of $|R|$ to which we subtract the distance to the corresponding extremum of the considered geometric parameter.

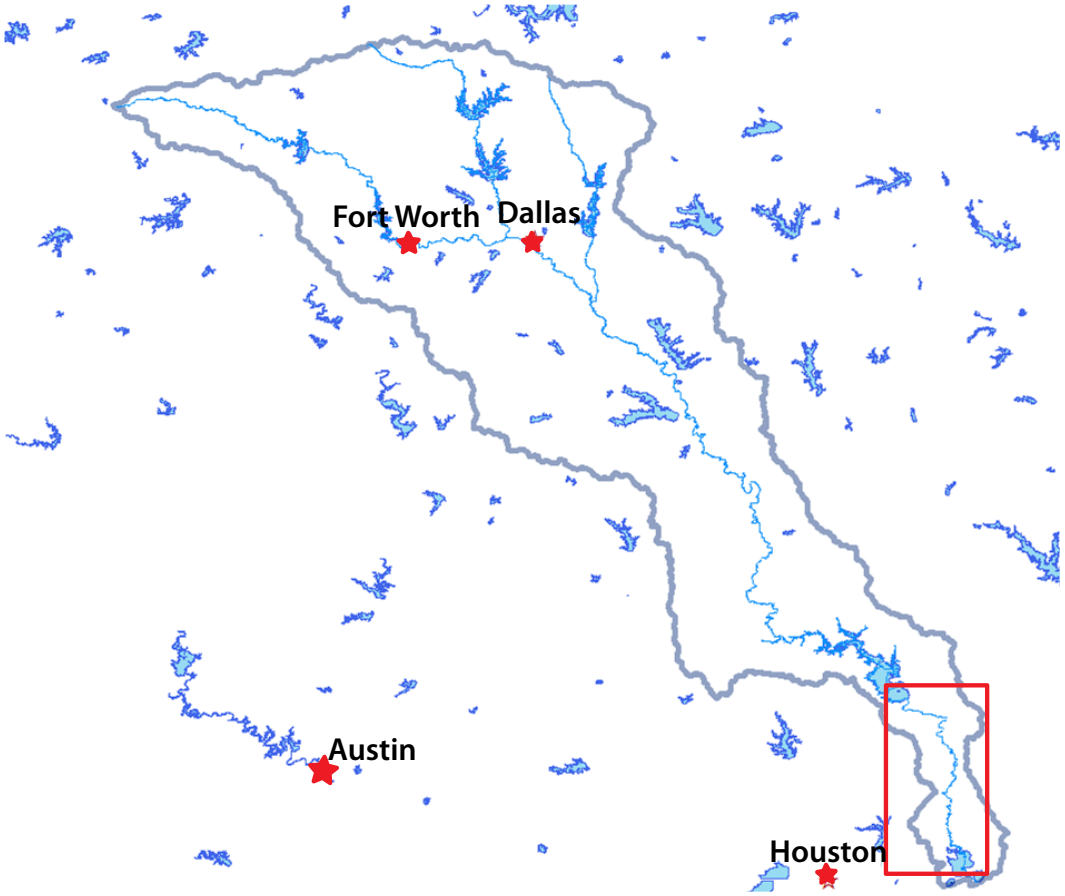


Figure 3.38: Trinity river and area of study in the red box.

(see the beginning of section 3.3.2.2 on page 76). An 11-point Ricker window was used to smooth the centerline data, and the "swipe length" was also 11 points. We color-coded plots to ouline spatial trends. The color code is given in Figure 3.39.

Radius of curvature and channel width values were subsequently matched using the same manual method as for the Mississippi. However, the matching process was done separately on three different parts of the river, as migration trends are different on each part (see figure 3.40 below):

- Section 1, from LLD to $\simeq$ 30 km downstream of LLD, is characterized by a low sinuosity and slow lateral migration due to the influence of the dam, possibly from sediment depletion [Williams and Wolman, 1984, Kondolf and Swanson, 1993];
- Section 2, from $\simeq$ 30 km to $\simeq$ 160 km downstream of LLD, is rapidly migrating and exhibits large sand bars; The average migration rate in this section is 2.3 $\mathrm{m} \cdot \mathrm{yr}^{-1}$, which represents 2% of the median width (116 m) per year;
- Section 3, from $\simeq$ 160 km downstream of LLD to the end of the dataset, is a backwater-influence zone that is migrating slowly [Hudson and Kesel, 2000], possibly due to changes in sediment transport [Nittrouer, 2010].

The results of the curvature-width analysis were also different for each section of the river:

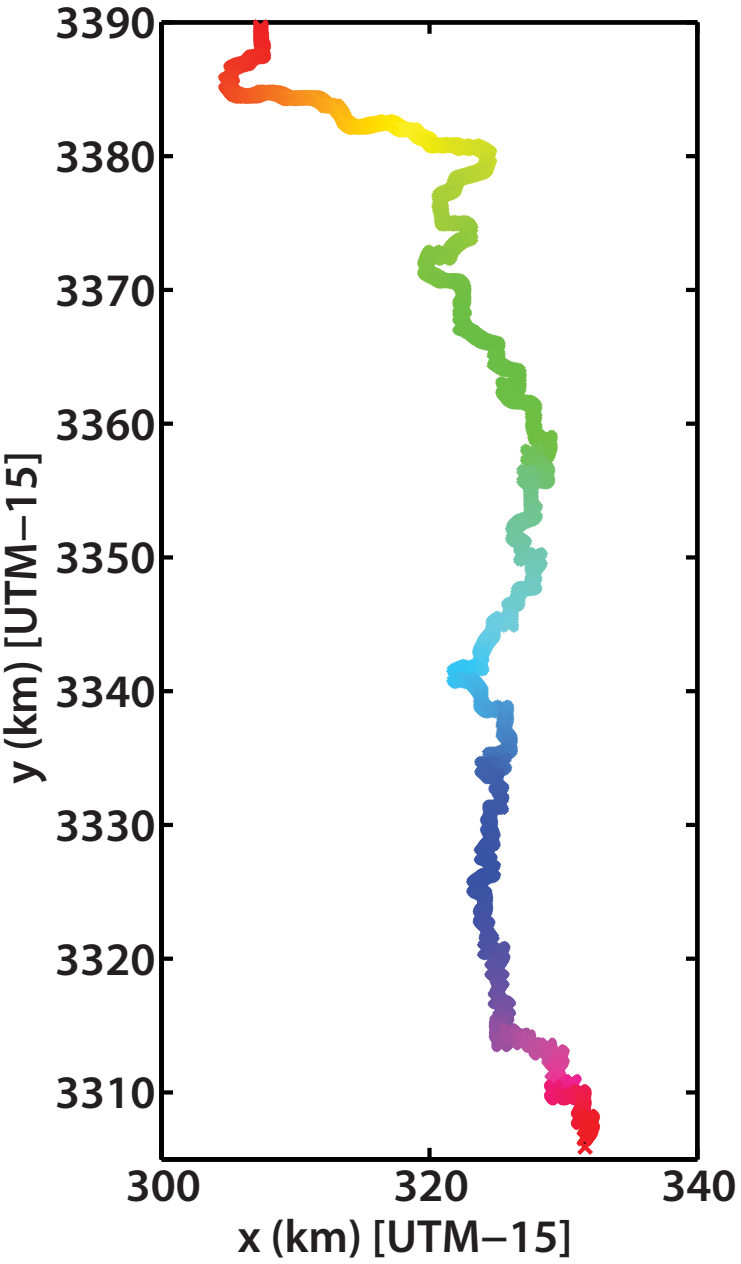


Figure 3.39: Color-coding of distance along Trinity river centerline: each point of the channel is uniquely associated with one color.

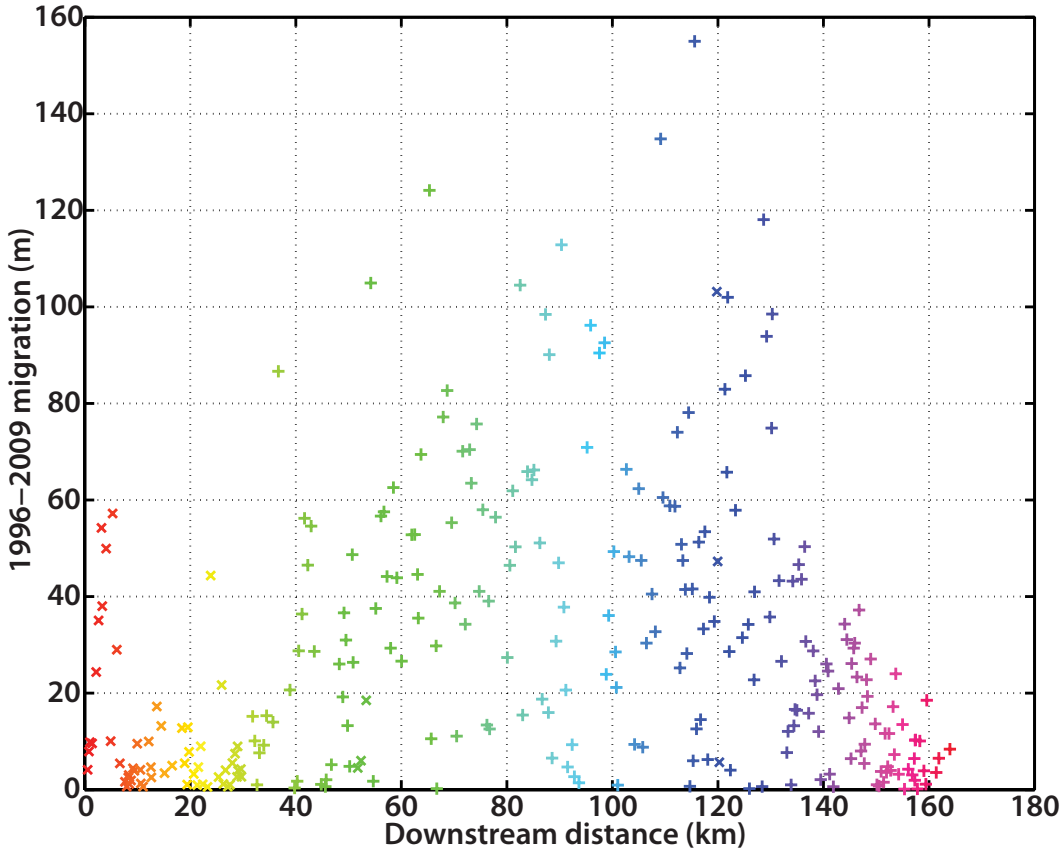


Figure 3.40: Migration rates of the Trinity river from 1996 to 2009 as a function of downstream distance from Lake Livingston Dam.

- In the dam-influenced section, radius of curvature and river width looked positively correlated ($|B| \simeq B_{0_p} \cdot |R|^{\eta_p}$, $\eta_p \simeq 0.037$); Positive correlation was also noticed at two locations in the second section:
  - the first location, corresponding to 3 centerline points located 0.5175 km, 0.5237 km and 0.5334 km downstream of LLD, presents a distinct change in cut/fill habits and migration rates compared to the rest of the second

section, and appears to be a point of transition for the height of the cut bank; however, this change is very gradual;

- the second location, corresponding to 3 centerline points located at 1.1978 km, 1.1995 km and 1.2026 km downstream of LLD had recently been cut off;

Points where positive *R*-*B* correlation was found are displayed in Figure 3.41 and the corresponding scatter-plot and least-square fit are given in Figure 3.42;

- In the rapidly-migrating section, radius of curvature and river width looked anti-correlated (negatively correlated; $|B| \simeq B_{0_n} \cdot |R|^{\eta_n}$, $\eta_n \simeq -0.064$); The color coding also shows that different parts of the rapidly-migrating section could be fit by similar relationships $|B| \simeq B_{0_i} \cdot |R|^{\eta_n}$ with the same slope η_n but different intercepts B_{0_i}; Points where negative *R*-*B* correlation was found are displayed in Figure 3.43 and the corresponding scatter plot and least-square fit are given in Figure 3.44;

- In the backwater zone, the trends were dampened and almost undistinguishable. Therefore, no correlation has been attempted in that area.

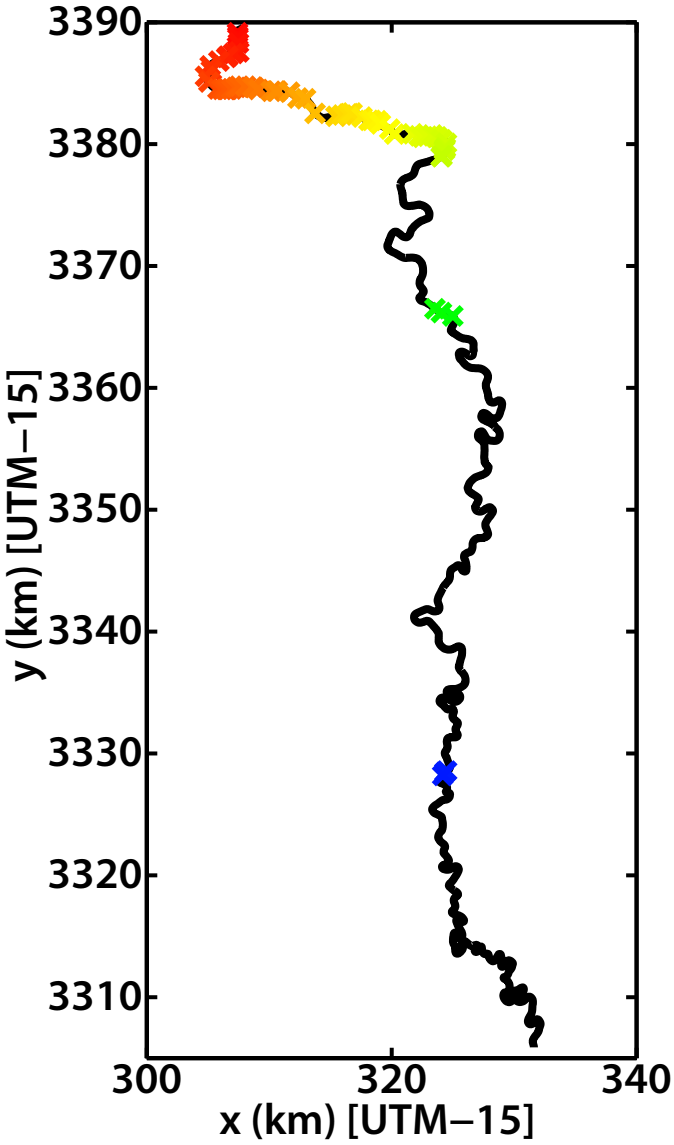


Figure 3.41: Areas of the Trinity river where positive correlation between the radius of curvature and the river width has been found.

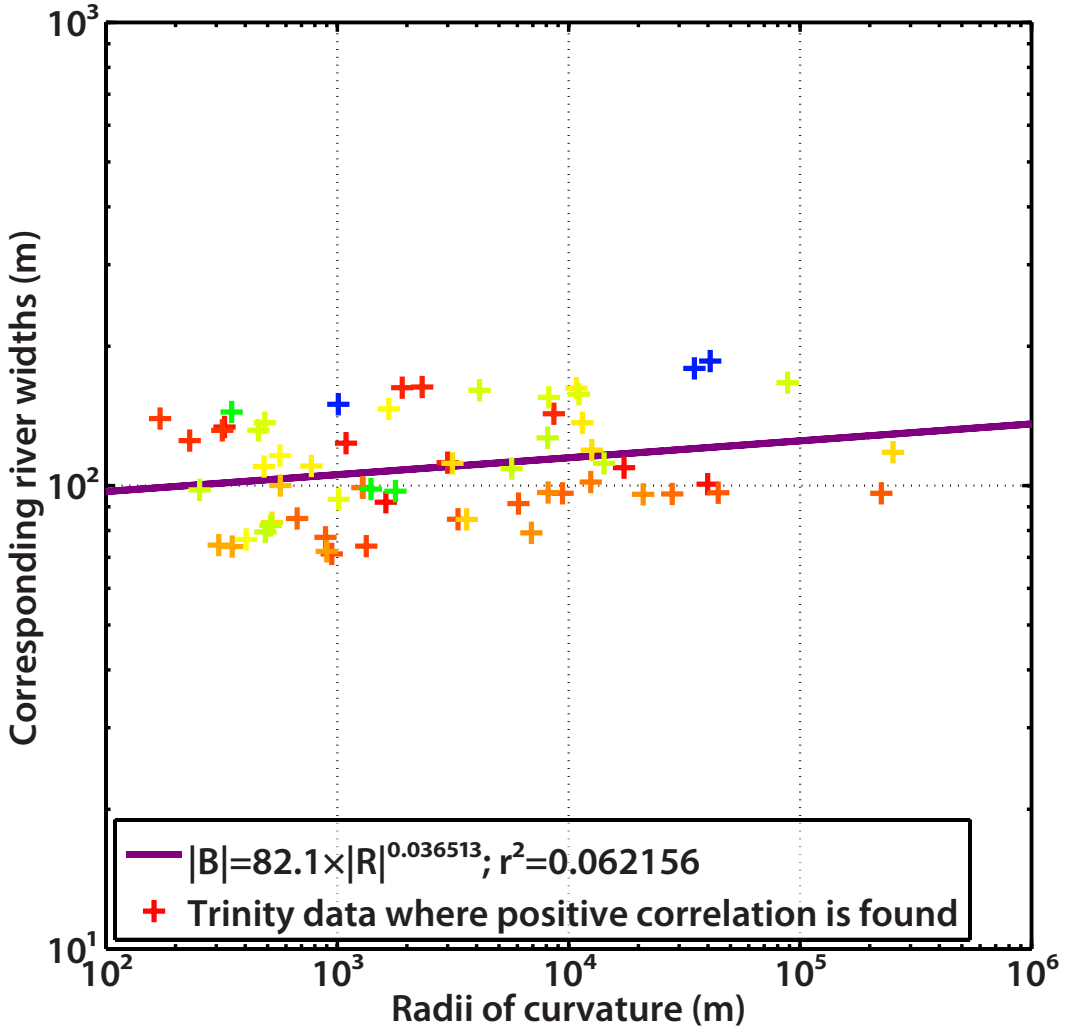


Figure 3.42: Trinity River width as a function of radius of curvature in the dam-influenced section. The colors of the plus signs correspond to the colors of points on Figure 3.41; The continuous purple line represents the associated least-square fit.

Although two examples such as the Mississippi and Trinity rivers do not allow us to make definitive conclusions, we hypothesize that rapidly-migrating rivers (such as the Trinity) will exhibit anti-correlations between radius of curvature and

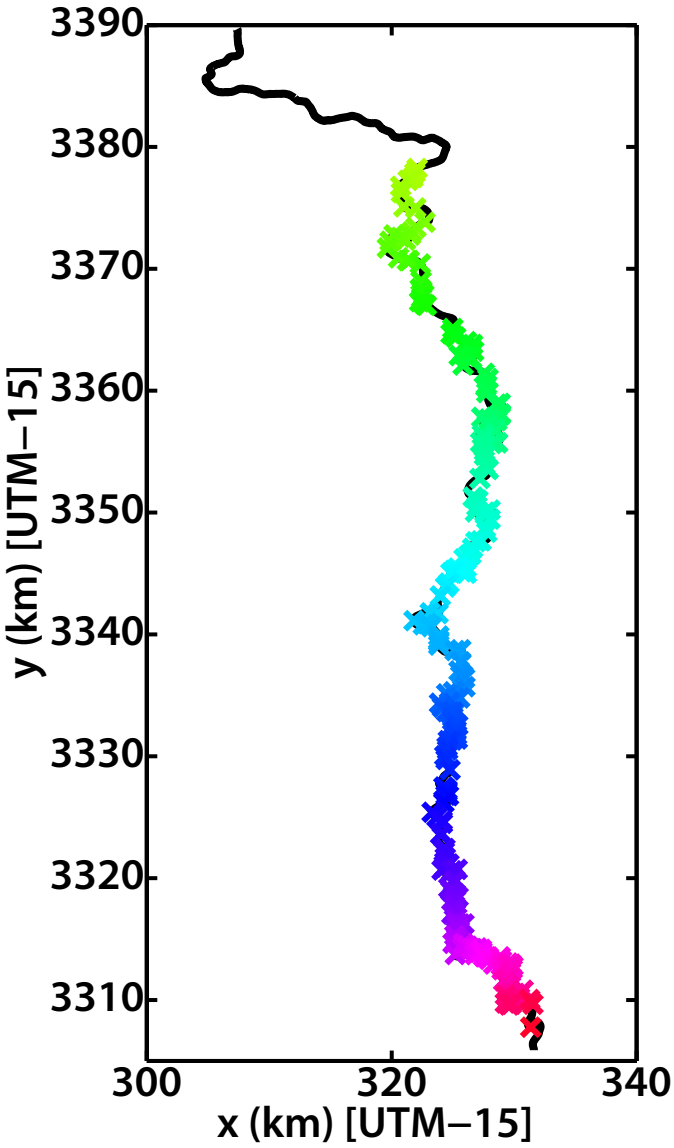


Figure 3.43: Areas of the Trinity river where anti-correlation (negative correlation) between the radius of curvature and the river width has been found.

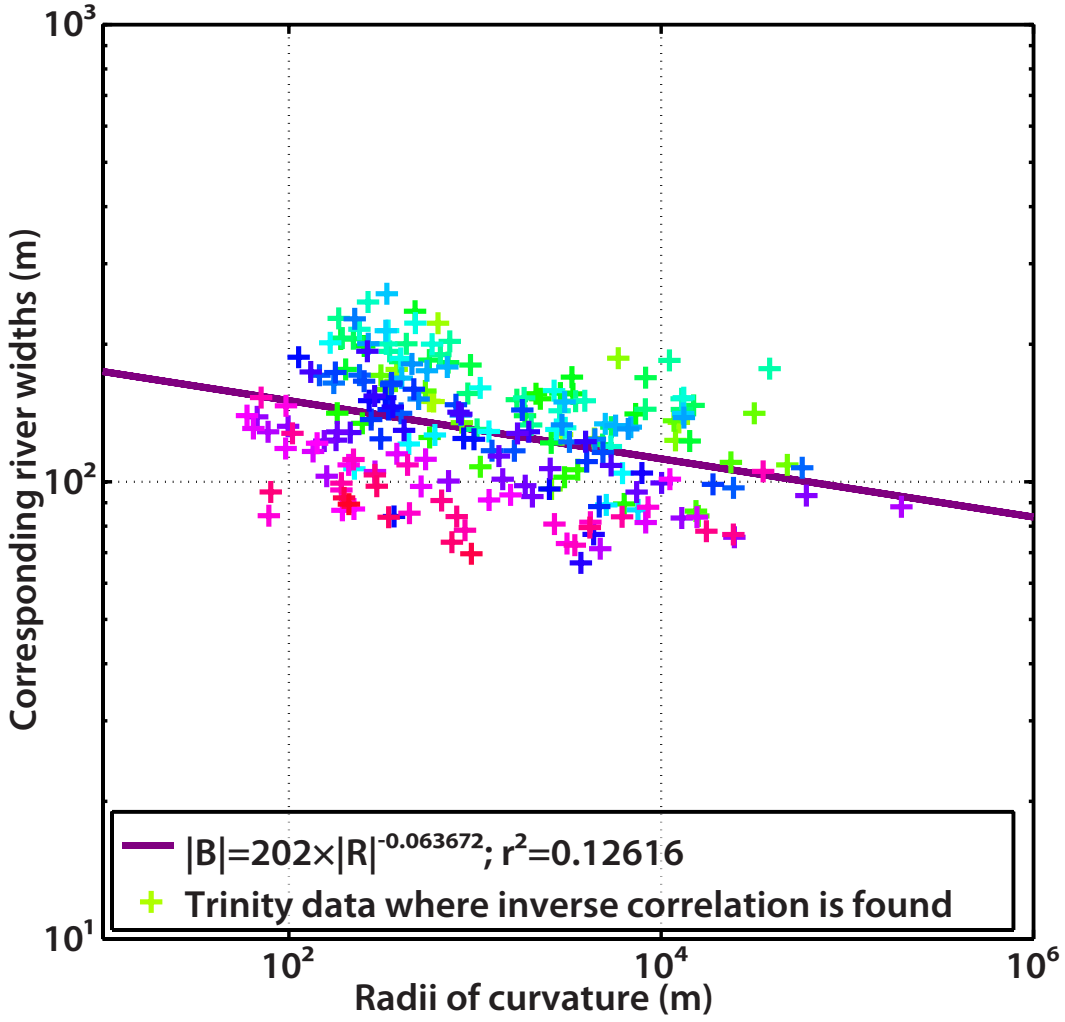


Figure 3.44: Trinity River width as a function of radius of curvature in the rapidly-migrating section. The colors of the plus signs correspond to the colors of points on Figure 3.43; The continuous purple line represents the associated least-square fit.

channel width, while slower-migrating rivers, such as the Mississippi or the first section of the Trinity, will show positively correlated values of radius of curvature and channel width.

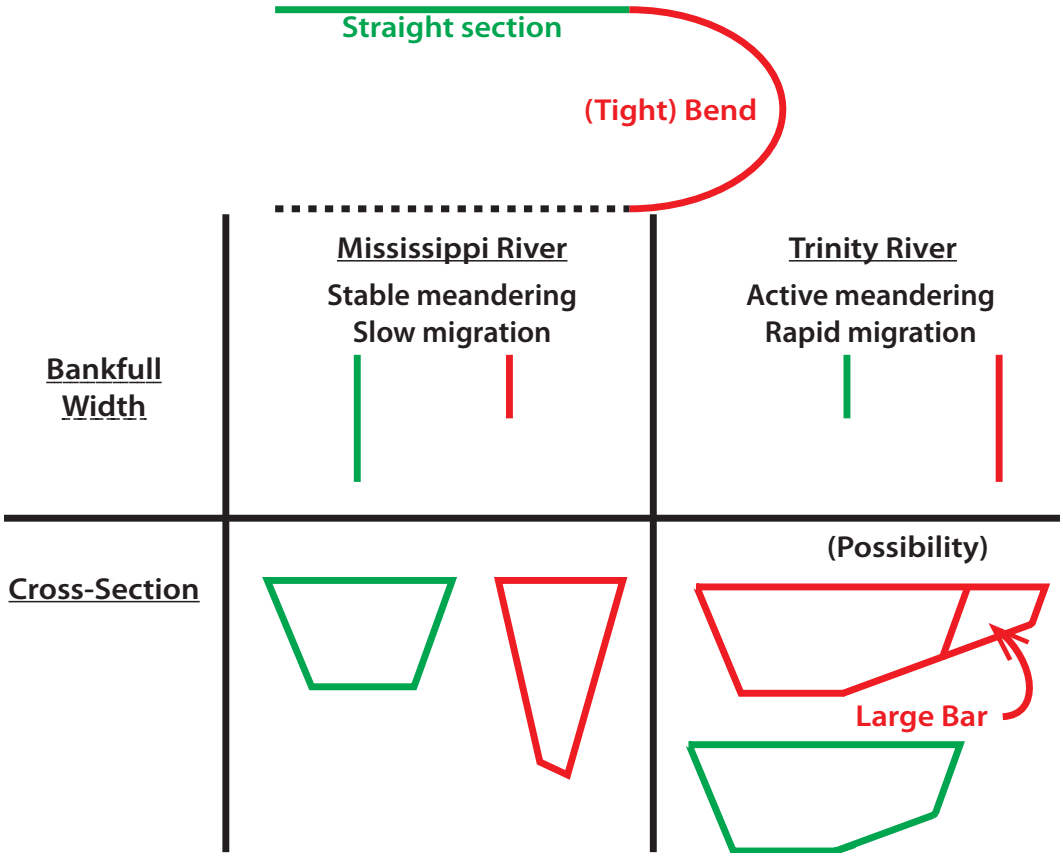


Figure 3.45: Summary of geometrical curvature-, width- and depth-trends in the Trinity and Mississippi rivers.

3.4 Conclusion

The recent availability of large datasets of channel coordinates, measured either directly on the field or indirectly through high-resolution orthophotographs,

and the increasing power of today's computers, is an occasion to begin automatic, large-scale analysis of geometric parameters of these channels. Our code, which computes radius of curvature and channel width as a function of curvilinear distance, is fast and accurate in most situations. Results from an analysis performed using this code on the Mississippi and Trinity rivers reveals that slowly-migrating rivers are wide and shallow in straight reaches, and deep and narrow in bends (Figure 3.45 summarizes these differences): migration happens by outer-bank pull; conversely, the bankfull width of rapidly-migrating rivers such as the Trinity river is larger in bends than in straight reaches, because of the presence of large point-bars: migration happens by bar push [Parker et al., 2011]. Therefore, the sign of the slope of the least-square fit line on a plot of the river width as a function of the radius of curvature (with logarithmic-scaled axes) may be an indicator of the river migration rate; in our analysis, this slope was positive for slowly migrating rivers and negative for rapidly-migrating rivers. While this process does not give a quantitative evaluation of the migration rate, it is a quick way to estimate qualitatively this migration rate. We hope that our code will be used on more datasets to confirm these first findings.

Conclusion

River channels, æolian dune fields and submarine channels differ by the type of transport that takes place in these environments: while transport in rivers and æolian dune fields happens primarily by bedload and the flow is close to capacity or at capacity, transport by turbidity currents in submarine channels is often far from capacity: erosion is localized but deposition may happen very far from the sediment input location. When the flow and bed interact actively, as is the case for bedload transport in river channels, finding analytical expressions for the co-evolution of bed and flow without simplifications is difficult, but we have shown in this dissertation that it is possible to correlate planform geometry with cross-sectional geometry, and to possibly predict migration rates; however, when the flow and the bed co-evolutions may be treated separately, as is the case for depositional turbidity currents or æolian transport), basic geometry allows us to model deposition very easily.

The qualitative results we found may also have direct applications for geologists or engineers:

- Provided the relationship relating channel curvature and channel bottom slope holds for other rivers, this relationship may be used to compute a compaction factor. If we suppose that the channel has not be laterally stretched or compressed, we can retrieve the original change in bottom level across the channel

from the radius of curvature and the bottom width. Comparing this original change in level with the change in level present in the rock record would yield the compaction factor for the studied area;

- If, once again, the relationship between width-curvature correlation and migration rate is valid for rivers other than the Trinity and the Mississippi, a set of digitized orthophotos is sufficient for estimating the mobility of channel bends. This migration rate is useful for all civil engineers building bridge piers or roads, who want to avoid problems with bank stability and erosion around the structures they build;

- Civil engineers designing a road (or other kinds of infrastructures) through a dune field may also want to be able to predict the state of that dune field after a given amount of time to prevent infrastructures from being buried. Being able to predict quickly how dunes will interact and move along a transect is a major asset in this case;

- Reservoir geologist can use the turbidity current code as an inverse model, proceeding from core measurements by trial and error to obtain the input conditions that would create a given distribution of grain sizes along a specific distance, then use the code again as a forward model to predict grain size patterns in the rest of the (uncored) reservoir.

I hope that the research presented in this dissertation will also stimulate more investigations into the various topics that I addressed. While the turbidity current and æolian modeling formulæ are derived from basic physics, the results that

I obtained with the curvature code are purely statistical. The correlation between curvature-width relationship and migration rate, which was demonstrated in three portions of river (Mississippi dataset, middle section of Trinity dataset and dam-influenced section of Trinity dataset) requires a particular attention, as 3 elements of correlation cannot be considered a general law of nature. Other correlations between curvature and cross-sectional geometrical parameters suffer from the same statistical problem, and the method I developed and followed needs to be applied to other rivers until a comprehensive number of correlations has been reached and the statistics can make sense.

Large datasets are needed for this kind of analysis and hopefully data collection of channel topography and shape will continue to increase. The æolian modeling code is also in a preliminary stage and while the objective of this code was to keep it as simple (and therefore efficient computationally) as possible, efficiency came at the cost of accuracy. We envision morphodynamics modeling in a multi-scale, multi-model framework; fine details of the flow require complex and computationally intensive codes, and could be applied to derive input rules for less intensive models of larger-scale phenomena. For instance, a full-flow model could be applied to two dunes to fine-tune the rules that we used for coalescence and repulsion. Eventually, the æolian code could be extended to a 2D model; However, the laws governing 1D interactions already need to be explained physically and need to be more accurate than the empirical parameters I used, and therefore a 2D model that contains interactions between side terminations of dunes would require an enormous amount of preliminary work. I also hope that the turbidity current

modeling can be extended, both physically through experiments and mathematically, to cases with low but non-zero entrainment. All the strengths and limitations of the work I presented in this dissertations stress the importance of geometry and statistics in geomorphology.

Appendices

Appendix A

Appendix to chapter 1

A.1 Derivation of Composite Advection Length formulæ

The formulæ in this section are derived for a surface that has slight irregularities but no large topography with respect to the flow height. We first assume that the global slope is 0 (between first and last points), i.e. that the original surface has been rotated. This rotated surface has coordinates (x_S, z_S). We assume that grains move forward at the velocity of the flow and move downwards at their respective settling velocities. In the interval of time that separates the deposition of two grains, we assume that the velocity of the flow is constant because it is only of function of sediment concentration (see section A.2 of this appendix); we also assume that in the same interval, the settling velocity of grains is constant, because the density of the current is constant. Therefore, during that interval of time, each grain has a downwards linear trajectory, and as a consequence, grains follow a piecewise linear trajectories between the initial time and their deposition time. For instance, the first grain to be deposited has forward velocity $U = U_0$, so that $x(t) = U \cdot t$, and downward settling velocity $V_1 = \frac{z_{1_i} \cdot U}{L_1(1)}$, hence its vertical position $z_1(t) = z_{1_i} - V_1 \cdot t = z_{1_i} \cdot \left[1 - \frac{U}{L_1(1)} \cdot t\right]$, and, with $t = \frac{x}{U}$, $z_1(t) = z_{1_i} \cdot \left[1 - \frac{x}{L_1(1)}\right]$, which is a linear trajectory.

A.1.1 Deposition of the first two grains

Following equation (A.1.1), the location of the deposition of the first grain is (x_{1_d}, z_{1_d}) such that x_{1_d} and z_{1_d} match equation (A.1.1) and $(x_{1_d}, z_{1_d}) \in (x_S, z_S)$.

$$z_1(t) = z_{1_i} \cdot \left[1 - \frac{x}{L_1(1)}\right] \tag{A.1.1}$$

Therefore, (x_{1_d}, z_{1_d}) is the solution to equation (A.1.2)

$$z_S = z_{1_i} \cdot \left[1 - \frac{x_S}{L_1(1)}\right] \tag{A.1.2}$$

or

$$L_1(1) \cdot \left[1 - \frac{z_S}{z_{1_i}}\right] = x_S \Longleftrightarrow x_S + \frac{L_1(1)}{z_{1_i}} \cdot z_S = L_1(1) \tag{A.1.3}$$

If there are multiple solutions (which may happen when the deposition surface is not flat), the chosen solution minimizes x_{1_d}. We also have $t_1 = \frac{x_{1_d}}{U_0}$.

Before grain 1 is deposited, grain 2 falls following equation (A.1.4)

$$z_2(t) = z_{1_i} \cdot \left[1 - \frac{x}{L_1(2)}\right] \tag{A.1.4}$$

Assuming that grain 2 does not get deposited before grain 1, the vertical position of grain 2 exactly when grain 1 is deposited is

$$z_2(t_1) = z_{1_i} \cdot \left[1 - \frac{x_{1_d}}{L_1(2)}\right] \tag{A.1.5}$$

Combining with equation (A.1.3)

$$z_2(t_1) = z_{1_i} \cdot \left[1 - \frac{L_1(1)}{L_1(2)} \cdot \left(1 - \frac{z_{1_d}}{z_{1_i}}\right)\right] = z_{1_i} + (z_{1_d} - z_{1_i}) \cdot \frac{L_1(1)}{L_1(2)} \tag{A.1.6}$$

For $t_1 \leq t \leq t_2$, the position of grain 2 is given by

$$\begin{aligned} z_2(t) &= z_2(t_1) - \frac{z_{2_i} \cdot (t - t_1) \cdot U_0}{L_2(2)} \\ &= z_{1_i} + (z_{1_d} - z_{1_i}) \cdot \frac{L_1(1)}{L_1(2)} - z_{2_i} \cdot \frac{(t - t_1) \cdot U_0}{L_2(2)} \end{aligned} \tag{A.1.7}$$

$$\begin{aligned} z_2(t) &= z_{1_i} + (z_{1_d} - z_{1_i}) \cdot \frac{L_1(1)}{L_1(2)} - z_{2_i} \cdot \frac{x - x_{1_d}}{L_2(2)} \\ &= z_{1_i} + (z_{1_d} - z_{1_i}) \cdot \frac{L_1(1)}{L_1(2)} - z_{2_i} \cdot \frac{x - L_1(1) \cdot \left[1 - \frac{z_{1_d}}{z_{1_i}}\right]}{L_2(2)} \end{aligned} \tag{A.1.8}$$

Therefore, (x_{2_d}, z_{2_d}) is the solution to equation (A.1.9)

$$z_S = z_{1_i} + (z_{1_d} - z_{1_i}) \cdot \frac{L_1(1)}{L_1(2)} + z_{2_i} \cdot \left(1 - \frac{z_{1_d}}{z_{1_i}}\right) \cdot \frac{L_1(1)}{L_2(2)} - \frac{z_{2_i}}{L_2(2)} \cdot x_S \tag{A.1.9}$$

or

$$x_S = \frac{z_{1_i} - z_S}{z_{2_i}} \cdot L_2(2) + \frac{z_{1_d} - z_{1_i}}{z_{2_i}} \cdot \frac{L_1(1)}{L_1(2)} \cdot L_2(2) + \left(1 - \frac{z_{1_d}}{z_{1_i}}\right) \cdot L_1(1) \tag{A.1.10}$$

Once again, if there are multiple solutions, the chosen solution minimizes x_{2_d}. We also have $t_2 = \frac{x_{2_d}}{U_0}$.

Before we go to grain 3, let's look if there is any condition which would have grain 2 deposited before grain 1. Before the first grain is deposited, we have

$$z_1(t) = z_{1_i} \cdot \left[1 - \frac{x(t)}{L_1(1)}\right] \tag{A.1.11}$$

and

$$z_2(t) = z_{1_i} \cdot \left[1 - \frac{x(t)}{L_1(2)}\right] \tag{A.1.12}$$

which means that

$$x(t) = L_1(1) \cdot \left[1 - \frac{z_1(t)}{z_{1_i}}\right] = L_1(2) \cdot \left[1 - \frac{z_2(t)}{z_{1_i}}\right] \tag{A.1.13}$$

i.e.

$$z_2(t) = z_{1_i} + [z_1(t) - z_{1_i}] \cdot \frac{L_1(1)}{L_1(2)} = z_1(t) \cdot \frac{L_1(1)}{L_1(2)} + z_{1_i} \cdot \left[1 - \frac{L_1(1)}{L_1(2)}\right] \tag{A.1.14}$$

At $t = 0$, $z_2(0) = z_{i_1} = z_1(0)$. For $t > 0$, $0 < \frac{L_1(1)}{L_1(2)} < 1$ Consequently, using $\frac{L_1(1)}{L_1(2)} = \xi, \quad \xi \in [0;1]$[1],

$$z_2(t) = \xi \cdot z_1(t) + (1 - \xi) \cdot z_{1_i} \tag{A.1.15}$$

which means that $z_2(t)$ is the barycenter of $(z_1(t), 1-\xi)$ and (z_{1_i}, ξ) and, given that $\xi > 0$, this barycenter is located between both heights $z_1(t)$ and z_{1_i}, i.e. $z_1(t) < z_2(t)\ \forall t \leq t_1$. Therefore, except if the topography somehow exhibits a cusp point, there should be no case where a grain i is deposited after a grain $i + j, j > 0$. Let's now derive the equation for grain 3. Figure 1.10 shows a simplified model of the deposition of grains 1, 2 and 3.

[1] Given that the virtual initial height is the same for $L_j(i)$ and $L_j(k)$, $j \neq k$, $\frac{L_j(i)}{L_j(k)} = \frac{w_s(i)}{w_s(k)}$

A.1.2 Deposition of the grains $3;\ldots;n$

Assuming that grain 3 gets deposited after grains 1 and 2, the vertical position of grain 3 for $t_0 \leq t \leq t_1$ is:

$$z_3(t) = z_3(t_0) - z_{1_i} \cdot \frac{x - x_{0_d}}{L_1(3)} \tag{A.1.16}$$

with $x_{0_d} = 0$ and $z_3(t_0) = z_j(t_0) = z_{1_i}\ \forall j \in \{1,\ldots,n\}$. For $t_1 \leq t \leq t_2$,

$$z_3(t) = z_3(t_1) - z_{2_i} \cdot \frac{x - x_{1_d}}{L_2(3)} \tag{A.1.17}$$

For $t_2 \leq t \leq t_3$,

$$z_3(t) = z_3(t_2) - z_{3_i} \cdot \frac{x - x_{2_d}}{L_3(3)} \tag{A.1.18}$$

i.e. at t_1, t_2 and t_3:

$$z_3(t_1) = z_3(t_0) - z_{1_i} \cdot \frac{x_{1_d} - x_{0_d}}{L_1(3)} \tag{A.1.19}$$

$$z_3(t_2) = z_3(t_1) - z_{2_i} \cdot \frac{x_{2_d} - x_{1_d}}{L_2(3)} \tag{A.1.20}$$

and

$$z_3(t_3) = z_3(t_2) - z_{3_i} \cdot \frac{x_{3_d} - x_{2_d}}{L_3(3)} \tag{A.1.21}$$

Combining these equations:

$$z_3(t_3) = z_3(t_0) - z_{1_i} \cdot \frac{x_{1_d} - x_{0_d}}{L_1(3)} - z_{2_i} \cdot \frac{x_{2_d} - x_{1_d}}{L_2(3)} - z_{3_i} \cdot \frac{x_{3_d} - x_{2_d}}{L_3(3)} \tag{A.1.22}$$

$$= z_3(t_0) - \frac{z_{1_i}}{L_1(3)} \cdot (x_{1_d} - x_{0_d}) - \frac{z_{2_i}}{L_2(3)} \cdot (x_{2_d} - x_{1_d}) - \frac{z_{3_i}}{L_3(3)} \cdot (x_{3_d} - x_{2_d}) \tag{A.1.23}$$

$$= z_3(t_0) + \left(\frac{z_{2_i}}{L_2(3)} - \frac{z_{1_i}}{L_1(3)} \right) \cdot x_{1_d} + \left(\frac{z_{3_i}}{L_3(3)} - \frac{z_{2_i}}{L_2(3)} \right) \cdot x_{2_d} - \frac{z_{3_i}}{L_3(3)} \cdot x_{3_d} \tag{A.1.24}$$

or

$$x_{3_d} = \frac{L_3(3)}{z_{3_i}} \cdot \left[z_3(t_0) - z_3(t_3) + \left(\frac{z_{2_i}}{L_2(3)} - \frac{z_{1_i}}{L_1(3)} \right) \cdot x_{1_d} + \left(\frac{z_{3_i}}{L_3(3)} - \frac{z_{2_i}}{L_2(3)} \right) \cdot x_{2_d} \right] \tag{A.1.25}$$

More generally, the equation of the trajectory of grain n is

$$\boxed{L_\star(n) = \frac{L_n(d_n)}{z_{n_i}} \cdot \left[z_n(t_0) - z_n(t_n) + \sum_{j=1}^{n-1} \left(\frac{z_{j+1_i}}{L_{j+1}(d_n)} - \frac{z_{j_i}}{L_j(d_n)} \right) \cdot L_\star(d_j) \right]} \tag{A.1.26}$$

$$\frac{L_\star(n)}{L_n(d_n)} = \frac{z_n(t_0) - z_n(t_n)}{z_{n_i}} + \sum_{j=1}^{n-1} \left(\frac{z_{j+1_i}/z_{n_i}}{L_{j+1}(d_n)} - \frac{z_{j_i}/z_{n_i}}{L_j(d_n)} \right) \cdot L_\star(d_j) \tag{A.1.27}$$

If $z_n(t_n) = 0$:

$$\frac{L_\star(n)}{L_n(d_n)} = \frac{z_{1_i}}{z_{n_i}} + \sum_{j=1}^{n-1} \left(\frac{z_{j+1_i}/z_{n_i}}{L_{j+1}(d_n)} - \frac{z_{j_i}/z_{n_i}}{L_j(d_n)} \right) \cdot L_\star(d_j) \tag{A.1.28}$$

This equation is solved at each step so that $(L_\star(n), z_n(t_n)) \in (x_S, z_S)$. We notice that the numbers $\gamma_n = \frac{L_n(d_n)}{z_{n_i}}$, $z_n(t_0) = z_{1_i}$, and $\delta_{j,n} = \frac{z_{j+1_i}}{L_{j+1}(d_n)} - \frac{z_{j_i}}{L_j(d_n)}$ may be computed and put in a table before beginning the recursive loop so that, using $\alpha_n = \gamma_n \cdot z_{1_i}$ and $\beta_{j,n} = \gamma_n \cdot \delta_{j,n}$, we have

$$L_\star(n) + \gamma_n \cdot z_n(t_n) = \alpha_n + \underbrace{\sum_{j=1}^{n-1} \beta_{j,n} \cdot L_\star(d_j)}_{\gamma_n \cdot \xi_n} \tag{A.1.29}$$

We define z_{j_i}, the "Virtual initial height"of all remaining grains at time t_j, as

$$z_{j_i} = \frac{\sum_{k=j}^{n} V_k \cdot Z_\star(k)}{\sum_{k=j}^{n} V_k} \tag{A.1.30}$$

where V_k is the volume of grain k.

A.2 Taking into account the changes in current velocity with deposition

We assume a constant densimetric Froude number Fr_d and look at the current depth-averaged velocities U_0 and U_1, where the grains of respective sizes m_0 and m_1 are the next grains to be deposited.

$$Fr_d = \frac{U}{\sqrt{\left(\frac{\rho_c}{\rho} - 1\right) \cdot g \cdot h}} \tag{A.2.1}$$

with

- $\rho_c = C \cdot \rho_s + (1-C) \cdot \rho_f$
- ρ_c: density of the current
- ρ_s: density of the sediment[2]
- ρ_f: density of the current fluid
- ρ: density of the ambient water
- C: Sediment concentration
- h: Current height
- g: Gravitational acceleration

Assuming that h is constant and $\rho_f = \rho$ allows us to simplify the relationship between U_0 and U_1:

$$\frac{U_1}{U_0} = \sqrt{\frac{\frac{\rho_{c_1}}{\rho} - 1}{\frac{\rho_{c_0}}{\rho} - 1}} = \sqrt{\frac{\rho_{c_1} - \rho}{\rho_{c_0} - \rho}} = \sqrt{\frac{C_1 \cdot (\rho_s - \rho)}{C_0 \cdot (\rho_s - \rho)}} = \sqrt{\frac{C_1}{C_0}} \tag{A.2.2}$$

The concentration of sediment is defined as the volume of grain sizes present at a certain time divided by a control volume V_c. If we assume that the grain sizes $d_1, d_2, \ldots, d_n$ are sorted from the largest grain (first to be deposited) to the smallest grain, i.e. $d_1 \geq d_2 \geq d_3 \geq \ldots \geq d_{n-1} \geq d_n$ (diameters of the grains), the grains remaining in the flow when grain m is the next one to be deposited are grains m, $m+1, \ldots, n$ and therefore:

$$C_m = \frac{\sum_{i=m}^{n} d_i^3}{V_c} \implies \frac{C_1}{C_0} = \frac{\sum_{i=m_1}^{n} d_i^3}{\sum_{i=m_0}^{n} d_i^3} \tag{A.2.3}$$

[2] or whatever is causing the excess weight of the current

which gives

$$\frac{U_1}{U_0} = \sqrt{\frac{\sum_{i=m_1}^{n} d_i^3}{\sum_{i=m_0}^{n} d_i^3}} \tag{A.2.4}$$

If U_0 is the initial velocity, then $m_0 = 1$ and

$$\frac{U_1}{U_0} = \sqrt{\frac{\sum_{i=m_1}^{n} d_i^3}{\sum_{i=1}^{n} d_i^3}} = f_1(U_0) \tag{A.2.5}$$

Given $p = \frac{w_s}{\kappa \cdot \sqrt{C_f} \cdot U}$, we get

$$p_1 = \frac{w_s}{\kappa \cdot \sqrt{C_f} \cdot U_1} = \frac{w_s}{\kappa \cdot \sqrt{C_f} \cdot U_0 \cdot f_1(U_0)} = \frac{p_0}{f_1(U_0)} \tag{A.2.6}$$

i.e.

$$p_1 = p_0 \cdot \sqrt{\frac{\sum_{i=1}^{n} d_i^3}{\sum_{i=m_1}^{n} d_i^3}} \tag{A.2.7}$$

Given the formula for $Z_\star$, there is no simple correlation between $Z_{\star_1}$ and $Z_{\star_0}$:

$$Z_\star = \frac{1-p}{2-p} \cdot \frac{\left(\frac{Z_a}{h}\right)^p - \left(\frac{Z_a}{h}\right)^2}{\left(\frac{Z_a}{h}\right)^p - \left(\frac{Z_a}{h}\right)} \cdot h = \frac{1-p}{2-p} \cdot \frac{\left(\frac{Z_a}{h}\right)^{p-2} - 1}{\left(\frac{Z_a}{h}\right)^{p-1} - 1} \cdot Z_a \tag{A.2.8}$$

yields

$$Z_{\star_1} = \frac{f_1(U_0) - p_0}{2 \cdot f_1(U_0) - p_0} \cdot \frac{\left(\frac{Z_a}{h}\right)^{\frac{p_0}{f_1(U_0)}-2} - 1}{\left(\frac{Z_a}{h}\right)^{\frac{p_0}{f_1(U_0)}-1} - 1} \cdot Z_a \tag{A.2.9}$$

When $p \to +\infty$ (full bedload), given that $\frac{Z_a}{h} < 1$,

$$Z_\star \to 1 \cdot \frac{0 - \left(\frac{Z_a}{h}\right)^2}{0 - \left(\frac{Z_a}{h}\right)} \cdot h \Longrightarrow Z_\star \to Z_a \tag{A.2.10}$$

When $p \to 0$ (full suspension, washload), given that $0 < \frac{Z_a}{h} \ll 1$,

$$Z_\star \to \frac{1}{2} \cdot \frac{1-0}{1-0} \cdot h \Longrightarrow Z_\star \to \frac{h}{2} \tag{A.2.11}$$

A.3 Experimental flow properties of García (1994)

Experimental flow properties of García's [1994] experiments are provided in table A.1.

Run	U_0 $(cm \cdot s^{-1})$	h_0 (cm)	C_0 (1)	R_{i_0} (1)	T_{inlet} (°C)	T_{flume} (°C)	Run time (min)
MIX1	13.3	3	3.64	0.10	17.0	16.5	45
MIX2	13.3	3	7.28	0.20	4.0	15.0	43
MIX3	13.3	3	7.28	0.20	7.5	7.5	34
MIX4	11.0	3	6.42	0.26	6.5	7.0	20
MIX5	11.0	3	7.28	0.29	6.0	7.0	30
MIX6	11.0	3	10.90	0.44	4.5	5.0	30
DEPO1	13.3	3	3.64	0.10	6.0	7.0	40
DEPO2	14.3	3	21.80	0.52	6.8	7.5	24
DEPO3	14.3	3	10.90	0.26	6.0	6.0	24

Table A.1: Input data for García's [1994] experiments

Note that the Richardson number R_i is the inverse of the squared densimetric Froude number.

A.4 Rouse profile

Assuming that downward motion of sediment is counteracted by upward eddy diffusion, we may write:

$$K_s(Z)\cdot\frac{\mathrm{d}C_s}{\mathrm{d}Z}=-w_s\cdot C_s\cdot(1-C_s)$$

Separating variables and integrating between Z_a and Z gives:

$$\frac{\mathrm{d}C_s}{C_s\cdot(1-C_s)}=-p\cdot\frac{\mathrm{d}Z}{Z}\Longrightarrow\left[\ln\left(\frac{C_s}{1-C_s}\right)\right]_{Z_a}^{Z}=-p\cdot[\ln(Z)]_{Z_a}^{Z}$$

i.e.

$$\frac{C_s(Z)}{1-C_s(Z)}=\frac{C_s(Z_a)}{1-C_s(Z_a)}\cdot\left(\frac{Z}{Z_a}\right)^{-p}$$

with:

- $K_s=\alpha\cdot\kappa\cdot u_\star\cdot Z$
- Z_a: Height of the bedload layer above bed
- h: flow depth
- Z: Height above bed
- $p=\dfrac{w_s}{\kappa\cdot u_\star}=\dfrac{w_s}{\kappa\cdot\sqrt{\dfrac{\tau_0}{\rho}}}$: Rouse number
- C_s: Sediment concentration
- C_a: Sediment concentration at $Z=Z_a$
- $Z'=Z-Z_a$, $h'=h-Z_a$
- $u_\star=\sqrt{C_f}\cdot\overline{u}$: Shear velocity
- $\overline{u}$: depth-averaged velocity
- $\kappa\simeq 0.407$: Von Kármán's constant
- C_f: Drag coefficient

Rouse [1937] proposed a slightly different formula describing the concen-

tration of sediment as a function of height:

$$\frac{C}{C_a} = \left[\left(\frac{1 - \frac{Z}{h}}{\frac{Z}{h}} \right) \cdot \left(\frac{\frac{Z_a}{h}}{1 - \frac{Z_a}{h}} \right) \right]^{\frac{w_s}{\kappa \cdot \sqrt{\frac{\tau_0}{\rho}}}} = \left[\frac{1 - \frac{Z'}{h'}}{1 - \frac{Z'}{Z_a}} \right]^{p} \tag{A.4.1}$$

Equation (A.4.1) is the equation used by Straub and Mohrig [2008] to compute the advection lengths of sediments being transported over levees in decelerating, depositional currents. Some examples of sediment concentration profiles produced by equation (A.4.1) are pictured in Figure A.1.

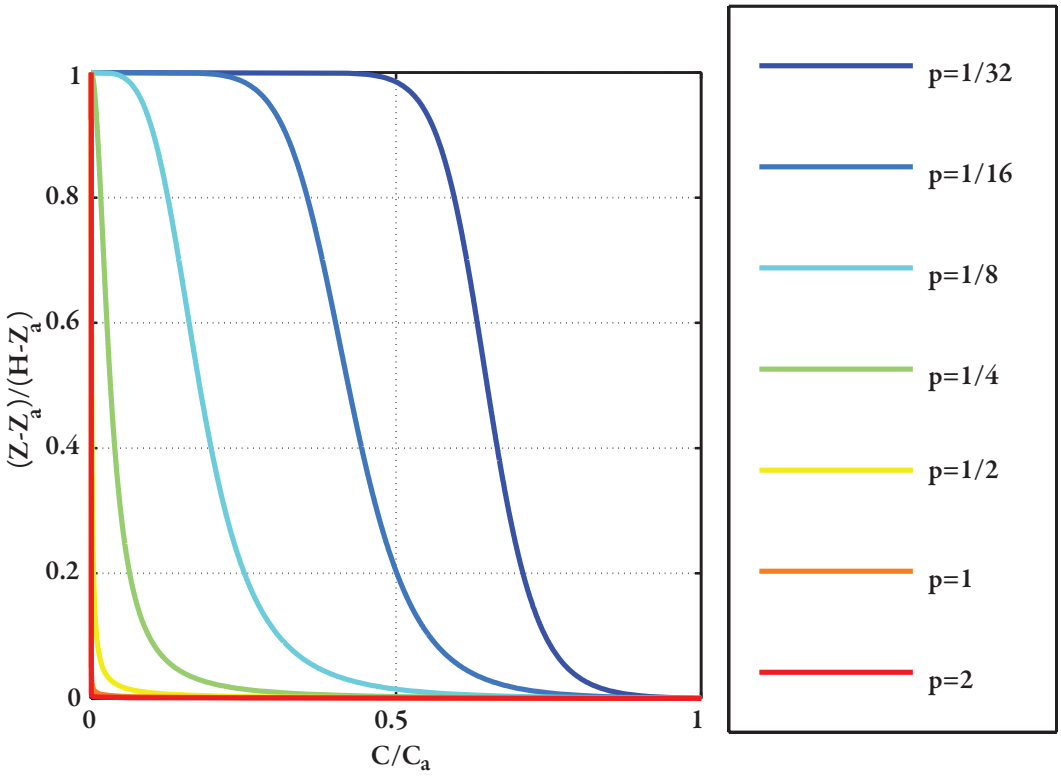


Figure A.1: Example of Rouse profiles

A.4.1 Center of mass of suspended sediment in a Rouse profile

If $C_s \ll 1$, then $\dfrac{1-C_s(Z)}{1-C_s(Z_a)} \simeq 1$ and in the lower flow $C_s(Z) \simeq C_s(Z_a) \cdot \left(\dfrac{Z_a}{Z}\right)^p$. Consequently,

$$Z_m = \frac{\int_{Z_a}^{h} C_s(Z) \cdot Z \, \mathrm{d}Z}{\int_{Z_a}^{h} C_s(Z) \, \mathrm{d}Z} \simeq \frac{1-p}{2-p} \cdot \frac{\left(\dfrac{Z_a}{h}\right)^p - \left(\dfrac{Z_a}{h}\right)^2}{\left(\dfrac{Z_a}{h}\right)^p - \dfrac{Z_a}{h}} \cdot h \qquad p \notin \{1,2\}$$

Therefore:

- if $p \to +\infty$ (pure bedload), then $Z_m = Z_a$;
- if $p \to 0$ (pure suspension, washload), then $Z_m = \frac{h}{2}$.

A.4.2 Characteristic distance above bed at which sediment distributed on a Rouse profile diffuses in the flow

Using the approximation $C_s(Z) \simeq C_s(Z_a) \cdot \left(\dfrac{Z_a}{Z}\right)^p$ in the lower flow,

$$\overline{C_s} \simeq \frac{1}{h} \cdot \int_{Z_a}^{h} C_s(Z) \, \mathrm{d}Z = \frac{C_s(Z_a)}{1-p} \cdot \left[\left(\frac{Z_a}{h}\right)^p - \frac{Z_a}{h}\right]$$

Hence the height associated with the average suspended sediment concentration:

$$C_s = \overline{C_s} \Longrightarrow C_s(Z_a) \cdot \left(\frac{Z_a}{Z_\star}\right)^p = \frac{C_s(Z_a)}{1-p} \cdot \left[\left(\frac{Z_a}{h}\right)^p - \frac{Z_a}{h}\right]$$

i.e.

$$Z_\star = \frac{Z_a}{\sqrt[p]{\dfrac{1}{1-p} \cdot \left[\left(\dfrac{Z_a}{h}\right)^p - \dfrac{Z_a}{h}\right]}} \qquad p \notin \{0,1\}$$

Appendix B

Appendix to chapter 2

B.1 Height-Velocity relationship

We assume that a dune ABC has moved forward by dx during a time dt, and is now as the position $A'B'C'$, as shown on Figure B.1.

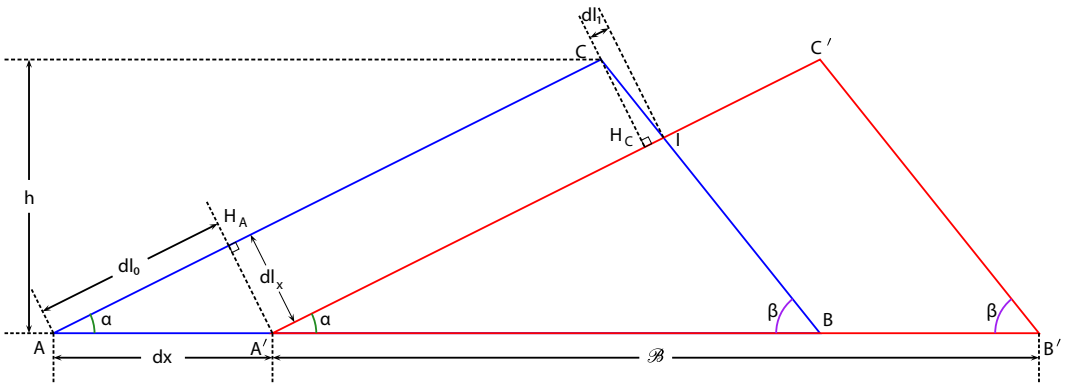


Figure B.1: Dune moving forward.

Assuming that $\widehat{ABC}$ is obtuse, the area d$\mathscr{A}$ that moved is the trapezoid

$AA'IC$ equal to:

$$\mathrm{d}\mathscr{A} = \left(AC + A'I\right) \cdot \frac{H_A A'}{2} \tag{B.1.1}$$

$$= \left(AC + AC - \mathrm{d}l_0 + \mathrm{d}l_1\right) \cdot \frac{\mathrm{d}l_x}{2} \tag{B.1.2}$$

$$= \left[2 \cdot AC - \mathrm{d}l_x \cdot \tan(\alpha) + \mathrm{d}l_x \cdot \tan\left(\widehat{H_C CI}\right)\right] \cdot \frac{\mathrm{d}l_x}{2} \tag{B.1.3}$$

$$= \left(2 \cdot AC + \mathrm{d}l_x \cdot \left[\tan\left(\widehat{H_C CI}\right) - \tan(\alpha)\right]\right) \cdot \frac{\mathrm{d}l_x}{2} \tag{B.1.4}$$

Given that $(AC) \parallel (A'C')$ and $(H_C C) \perp (A'C')$, $(AC) \parallel (H_C C)$. Consequently, $\widehat{H_C CI} = \widehat{ACI} - \frac{\pi}{2}$ with $\widehat{ACI} = \widehat{ACB} = \pi - (\alpha + \beta)$. Therefore, $\widehat{H_C CI} = \frac{\pi}{2} - (\alpha + \beta)$ and $\tan\left(\widehat{H_C CI}\right) = \tan\left[\frac{\pi}{2} - (\alpha + \beta)\right] = \dfrac{1}{\tan(\alpha + \beta)}$. With $\mathrm{d}l_x = \mathrm{d}x \cdot \sin(\alpha)$ and $AC = \frac{h}{\sin(\alpha)}$:

$$\mathrm{d}\mathscr{A} = \left(\frac{2 \cdot h}{\sin(\alpha)} + \mathrm{d}x \cdot \sin(\alpha) \cdot \left[\frac{1}{\tan(\alpha + \beta)} - \tan(\alpha)\right]\right) \cdot \frac{\mathrm{d}x \cdot \sin(\alpha)}{2} \tag{B.1.5}$$

$$= h \cdot \mathrm{d}x + \underbrace{\mathrm{d}x^2 \cdot \frac{\sin^2(\alpha)}{2} \left[\frac{1 - 2 \cdot \tan(\alpha) \cdot \tan(\beta) - \tan^2(\alpha)}{\tan(\alpha) + \tan(\beta)}\right]}_{\text{Second-order term} \simeq 0} \tag{B.1.6}$$

$$\simeq h \cdot \mathrm{d}x \tag{B.1.7}$$

Therefore, if v is the velocity of the moving dune,

$$\frac{\mathrm{d}\mathscr{A}}{\mathrm{d}t} \simeq h \cdot \frac{\mathrm{d}x}{\mathrm{d}t} \tag{B.1.8}$$

$$\simeq h \cdot v \tag{B.1.9}$$

If we assume that the wind can transport a given, constant sediment (mass) flux whatever the size of the dune, then $\dfrac{\mathrm{d}\mathscr{A}}{\mathrm{d}t} = \text{constant} = \phi$ implies $v = \dfrac{\phi}{h}$. That is, the velocity of the dune is inversely proportional to its height.

B.2 Proportionality between upwind dune size and size of the ejected dune during a repulsion

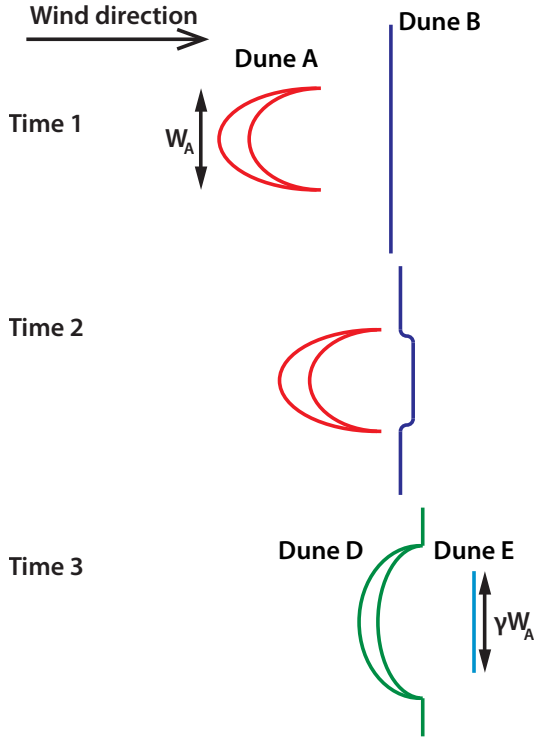


Figure B.2: Planform view of idealized steps of the repulsion process between a parabolic dune and a straight-crested dune.

We provide the planview of an idealized repulsion process in Figure B.2. Although the code is only uni-dimensional, we have to consider the width W of a dune in this section. We assume that an upwind parabolic dune $\mathcal{A}$ of width $W_{\mathcal{A}}$ is colliding with a downwind straight-crested dune $\mathcal{B}$ of width $W_{\mathcal{B}}$ and that a repulsion will happen; the dunes are perfectly aligned. We also assume that the width of a dune is proportional to its height: $W_{\mathcal{A}} = \delta \cdot h_{\mathcal{A}}$ and $W_{\mathcal{B}} = \delta \cdot h_{\mathcal{B}}$. During the repulsion process (Time 2 in Figure B.2, a central section of dune $\mathcal{B}$ is pushed forward by the

advancing Dune $\mathcal{A}$. The width of this section is proportional to the width of dune $\mathcal{A}$. This section slowly detaches from Dune $\mathcal{B}$ to form a new dune $\mathcal{E}$, while the extremities of dune $\mathcal{A}$ are attaching to the remaining parts of dune $\mathcal{B}$ to form dune $\mathcal{D}$ (Time 3 on Figure B.2). The width of dune $\mathcal{E}$ is therefore proportional to that of dune $\mathcal{A}$: $W_{\mathcal{E}} = \gamma \cdot W_{\mathcal{A}}$. The assumption that $W_{\mathcal{A}} = \delta \cdot h_{\mathcal{A}}$ and $W_{\mathcal{E}} = \delta \cdot h_{\mathcal{E}}$ implies that $h_{\mathcal{A}} = (1/\delta \cdot h_{\mathcal{A}}) \cdot \gamma \cdot \delta$, i.e. the height of the ejected dune is proportional to the height of the original upwind dune.

B.3 Height relationships during interactions

We translate the conservation of dune mass during interactions as the conservation of triangle surface. For the interaction of an upwind dune $\mathcal{A}$ and a downwind dune $\mathcal{B}$ that results either in a single dune $\mathcal{C}$ (coalescence) or two dunes $\mathcal{D}$ and $\mathcal{E}$, we have:

- Area of a dune:

$$\mathscr{A} = \frac{b}{2} \cdot h \tag{B.3.1}$$

  where b is the base length (footprint) and h the height. If R_{bh} is the base-to-height ratio defined in section 2.2,

$$\mathscr{A} = \frac{h \cdot R_{bh}}{2} \cdot h = h^2 \cdot \frac{R_{bh}}{2} \tag{B.3.2}$$

- The conservation of mass is therefore written

$$\mathscr{A}_{\mathcal{A}} + \mathscr{A}_{\mathcal{B}} = \mathscr{A}_{\mathcal{C}} = \mathscr{A}_{\mathcal{D}} + \mathscr{A}_{\mathcal{E}} \tag{B.3.3}$$

i.e.

$$h_{\mathcal{A}}^2 \cdot \frac{R_{bh}}{2} + h_{\mathcal{B}}^2 \cdot \frac{R_{bh}}{2} = h_{\mathcal{C}}^2 \cdot \frac{R_{bh}}{2} = h_{\mathcal{D}}^2 \cdot \frac{R_{bh}}{2} + h_{\mathcal{E}}^2 \cdot \frac{R_{bh}}{2} \tag{B.3.4}$$

and therefore

$$h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2 = h_{\mathcal{C}}^2 = h_{\mathcal{D}}^2 + h_{\mathcal{E}}^2 \tag{B.3.5}$$

For the case of a repulsion, the transfer of height between $\mathcal{A}$ and $\mathcal{D}$ is

$$h_{\mathcal{D}} = h_{\mathcal{B}} + p_1 \cdot h_{\mathcal{A}} = h_{\mathcal{A}} \cdot \left(\frac{h_{\mathcal{B}}}{h_{\mathcal{A}}} + p_1 \right) \tag{B.3.6}$$

Conbining equations (B.3.5) and (B.3.6) yields the equation for the height of dune $\mathcal{E}$

$$h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2 = (h_{\mathcal{B}} + p_1 \cdot h_{\mathcal{A}})^2 + h_{\mathcal{E}}^2 \tag{B.3.7}$$

$$h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2 = h_{\mathcal{B}}^2 + p_1^2 \cdot h_{\mathcal{A}}^2 + 2 \cdot p_1 \cdot h_{\mathcal{A}} \cdot h_{\mathcal{B}} + h_{\mathcal{E}}^2 \tag{B.3.8}$$

$$h_{\mathcal{E}}^2 = h_{\mathcal{A}}^2 - \left(p_1^2 \cdot h_{\mathcal{A}}^2 + 2 \cdot p_1 \cdot h_{\mathcal{A}} \cdot h_{\mathcal{B}} \right) \tag{B.3.9}$$

$$h_{\mathcal{E}}^2 = h_{\mathcal{A}}^2 \cdot \left[1 - p_1^2 - 2 \cdot p_1 \cdot \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}} \right] \tag{B.3.10}$$

$$h_{\mathcal{E}} = h_{\mathcal{A}} \cdot \sqrt{1 - p_1^2 - 2 \cdot p_1 \cdot \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}} \tag{B.3.11}$$

- The square root prevents some combinations of p_1 and $\frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}$ from giving real results. Using the notation $R_{\mathcal{B}\mathcal{A}} = \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}$, the condition of existence given by equation (B.3.11) is

$$1 - p_1^2 - 2 \cdot p_1 \cdot R_{\mathcal{B}\mathcal{A}} \geqslant 0 \tag{B.3.12}$$

$$1 - p_1^2 \geqslant 2 \cdot p_1 \cdot R_{\mathcal{B}\mathcal{A}} \tag{B.3.13}$$

$$R_{\mathcal{B}\mathcal{A}} \leqslant \frac{1 - p_1^2}{2 \cdot p_1} \tag{B.3.14}$$

$$\tag{B.3.15}$$

Therefore, p_1 and the maximum merge ratio *mmr cannot* be chosen independently; a required condition for the code to continue without encountering issues created by imaginary numbers is:

$$\frac{2 \cdot p_1}{1 - p_1^2} \leqslant mmr \tag{B.3.16}$$

or

$$0 < p_1 \leq \frac{\sqrt{1 + mmr^2} - 1}{mmr}, \quad 0 \leq mmr < 1 \tag{B.3.17}$$

This equation implies that in any case, p_1 may not be larger than $\sqrt{2} - 1 \simeq 0.4142$, and that it is wiser to first choose *mmr* then p_1.

B.4 Positions of the dunes after interaction

The positions of the center of mass does not change through an interaction. Therefore, the new center of mass (or the barycenter of the new centers of mass) is equal to the barycenter of the old centers of mass with the weights equal to the square heights.

B.4.1 Coalescence

For the coalescence case, using the same notations as for the previous section:

$$h_{\mathcal{A}}^2 \cdot x_{G_{\mathcal{A}}} + h_{\mathcal{B}}^2 \cdot x_{G_{\mathcal{A}}} = h_{\mathcal{C}}^2 \cdot x_{G_{\mathcal{C}}} \tag{B.4.1}$$

Per Thales's theorem, the abscissa of the center of mass (centroid of the triangle) is located at one third of the distance between the base mid-point K and

the foot of the altitude H:

$$x_G = x_K + \frac{x - x_K}{3} \tag{B.4.2}$$

With $x_K = \frac{x_{tail} + x_{head}}{2}$, $x_{tail} = x - \frac{h}{\tan(\alpha)}$ and $x_{head} = x + \frac{h}{\tan(\beta)}$, we obtain

$$x_G = \frac{x_{tail} + x_{head}}{2} + \frac{1}{3} \cdot \left(x - \frac{x_{tail} + x_{head}}{2} \right) \tag{B.4.3}$$

$$x_G = x + h \cdot \underbrace{\frac{1}{3} \cdot \left[-\frac{1}{\tan(\alpha)} + \frac{1}{\tan(\beta)} \right]}_{\mathcal{F}_1} \tag{B.4.4}$$

$$x_G = x + h \cdot \mathcal{F}_1 \tag{B.4.5}$$

Consequently,

$$h_{\mathcal{A}}^2 \cdot (x_{\mathcal{A}} + h_{\mathcal{A}} \cdot \mathcal{F}_1) + h_{\mathcal{B}}^2 \cdot (x_{\mathcal{B}} + h_{\mathcal{B}} \cdot \mathcal{F}_1) = h_{\mathcal{C}}^2 \cdot (x_{\mathcal{C}} + h_{\mathcal{C}} \cdot \mathcal{F}_1) \tag{B.4.6}$$

Combine with equation (B.3.5)

$$h_{\mathcal{A}}^2 \cdot (x_{\mathcal{A}} + h_{\mathcal{A}} \cdot \mathcal{F}_1) + h_{\mathcal{B}}^2 \cdot (x_{\mathcal{B}} + h_{\mathcal{B}} \cdot \mathcal{F}_1) = \left(h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2 \right) \cdot x_{\mathcal{C}} + \left(h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2 \right)^{3/2} \cdot \mathcal{F}_1 \tag{B.4.7}$$

i.e.

$$x_{\mathcal{C}} = \frac{h_{\mathcal{A}}^2 \cdot x_{\mathcal{A}} + h_{\mathcal{B}}^2 \cdot x_{\mathcal{B}} + \mathcal{F}_1 \cdot \left[h_{\mathcal{A}}^3 + h_{\mathcal{B}}^3 - \left(h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2 \right)^{3/2} \right]}{h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2} \tag{B.4.8}$$

Simplifying by $h_{\mathcal{A}}^2$ and using $R_{\mathcal{B}\mathcal{A}} = \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}$:

$$x_{\mathcal{C}} = \frac{x_{\mathcal{A}} + R_{\mathcal{B}\mathcal{A}}^2 \cdot x_{\mathcal{B}} + h_{\mathcal{A}} \cdot \mathcal{F}_1 \cdot \left[1 + R_{\mathcal{B}\mathcal{A}}^3 - \left(1 + R_{\mathcal{B}\mathcal{A}}^2 \right)^{3/2} \right]}{1 + R_{\mathcal{B}\mathcal{A}}^2} \tag{B.4.9}$$

B.4.2 Repulsion

Using, again, the same notations as for the previous section:

$$h_{\mathcal{A}}^2 \cdot x_{G_{\mathcal{A}}} + h_{\mathcal{B}}^2 \cdot x_{G_{\mathcal{A}}} = h_{\mathcal{D}}^2 \cdot x_{G_{\mathcal{D}}} + h_{\mathcal{E}}^2 \cdot x_{G_{\mathcal{E}}} \tag{B.4.10}$$

The condition for the head of the new upwind dune to touch the tail of the new downwind dune is

$$x_{\mathcal{D}} + \frac{h_{\mathcal{D}}}{\tan(\alpha)} = x_{\mathcal{E}} - \frac{h_{\mathcal{E}}}{\tan(\beta)} \tag{B.4.11}$$

i.e.

$$x_{\mathcal{E}} = x_{\mathcal{D}} + \frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)} \tag{B.4.12}$$

Also, after combining with equation (B.4.5), the conservation of the center of mass can be written:

$$x_{\mathcal{D}} \cdot h_{\mathcal{D}}^2 + x_{\mathcal{E}} \cdot h_{\mathcal{E}}^2 + \left(h_{\mathcal{D}}^3 + h_{\mathcal{E}}^3\right) \cdot \mathcal{F}_1 = x_{\mathcal{A}} \cdot h_{\mathcal{A}}^2 + x_{\mathcal{B}} \cdot h_{\mathcal{B}}^2 + \left(h_{\mathcal{A}}^3 + h_{\mathcal{B}}^3\right) \cdot \mathcal{F}_1 \tag{B.4.13}$$

for the updated model.

Partially combining with equations (B.3.6), (B.3.11) and (B.4.11):

$$\begin{aligned} x_{\mathcal{D}} \cdot (h_{\mathcal{B}} + p_1 \cdot h_{\mathcal{A}})^2 + \left(x_{\mathcal{D}} + \frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot \left(1 - p_1^2 - 2 \cdot p_1 \cdot \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right) \cdot h_{\mathcal{A}}^2 = \\ = x_{\mathcal{A}} \cdot h_{\mathcal{A}}^2 + x_{\mathcal{B}} \cdot h_{\mathcal{B}}^2 + \left(h_{\mathcal{A}}^3 + h_{\mathcal{B}}^3 - h_{\mathcal{D}}^3 - h_{\mathcal{E}}^3\right) \cdot \mathcal{F}_1 \end{aligned} \tag{B.4.14}$$

$$x_{\mathcal{D}} \cdot \left(h_{\mathcal{A}}^2 + h_{\mathcal{B}}^2\right) + h_{\mathcal{A}}^2 \cdot \left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot \left(1 - p_1^2 - 2 \cdot p_1 \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right) = \\ = x_{\mathcal{A}} \cdot h_{\mathcal{A}}^2 + x_{\mathcal{B}} \cdot h_{\mathcal{B}}^2 + \left(h_{\mathcal{A}}^3 + h_{\mathcal{B}}^3 - h_{\mathcal{D}}^3 - h_{\mathcal{E}}^3\right) \cdot \mathcal{F}_1 \tag{B.4.15}$$

Dividing by $h_{\mathcal{A}}^2 \neq 0$:

$$x_{\mathcal{D}} \cdot \left[1 + \left(\frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right)^2\right] + \left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot \left(1 - p_1^2 - 2 \cdot p_1 \cdot \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right) = x_{\mathcal{A}} + x_{\mathcal{B}} \cdot \left(\frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right)^2 + \ldots \\ \ldots + h_{\mathcal{A}} \cdot \left[1 - 3 \cdot p_1 \cdot \left(\frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right)^2 - 3 \cdot \left(\frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right) \cdot p_1^2 - p_1^3 - \left(1 - p_1^2 - 2 \cdot p_1 \cdot \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}\right)^{\frac{3}{2}}\right] \cdot \mathcal{F}_1 \tag{B.4.16}$$

Using again the reduced variable $R_{\mathcal{B}\mathcal{A}} = \frac{h_{\mathcal{B}}}{h_{\mathcal{A}}}$ to rewrite this equation, we obtain:

$$x_{\mathcal{D}} \cdot \left[1 + R_{\mathcal{B}\mathcal{A}}^2\right] + \left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot \left[1 + R_{\mathcal{B}\mathcal{A}}^2 - (R_{\mathcal{B}\mathcal{A}} + p_1)^2\right] = x_{\mathcal{A}} + x_{\mathcal{B}} \cdot r^2 + \ldots \\ \ldots h_{\mathcal{A}} \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - (R_{\mathcal{B}\mathcal{A}} + p_1)^3 - \left[1 + R_{\mathcal{B}\mathcal{A}}^2 - (R_{\mathcal{B}\mathcal{A}} + p_1)^2\right]^{\frac{3}{2}}\right) \cdot \mathcal{F}_1 \tag{B.4.17}$$

which means:

$$
\begin{aligned}
x_{\mathcal{D}} = \frac{x_{\mathcal{A}} + x_{\mathcal{B}} \cdot R^2_{\mathcal{B}\mathcal{A}}}{1 + R^2_{\mathcal{B}\mathcal{A}}} + \ldots \\
\ldots + h_{\mathcal{A}} \cdot \mathcal{F}_1 \cdot \frac{1 + R^3_{\mathcal{B}\mathcal{A}} - (R_{\mathcal{B}\mathcal{A}} + p_1)^3 - \left[1 + R^2_{\mathcal{B}\mathcal{A}} - (R_{\mathcal{B}\mathcal{A}} + p_1)^2\right]^{\frac{3}{2}}}{1 + R^2_{\mathcal{B}\mathcal{A}}} + \ldots \\
\ldots - \left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot \frac{1 + R^2_{\mathcal{B}\mathcal{A}} - (R_{\mathcal{B}\mathcal{A}} + p_1)^2}{1 + R^2_{\mathcal{B}\mathcal{A}}}
\end{aligned}
\tag{B.4.18}
$$

and

$$
\begin{aligned}
x_{\mathcal{E}} = \frac{x_{\mathcal{A}} + x_{\mathcal{B}} \cdot R^2_{\mathcal{B}\mathcal{A}}}{1 + R^2_{\mathcal{B}\mathcal{A}}} + \ldots \\
\ldots + h_{\mathcal{A}} \cdot \mathcal{F}_1 \cdot \frac{1 + R^3_{\mathcal{B}\mathcal{A}} - (R_{\mathcal{B}\mathcal{A}} + p_1)^3 - \left[1 + R^2_{\mathcal{B}\mathcal{A}} - (R_{\mathcal{B}\mathcal{A}} + p_1)^2\right]^{\frac{3}{2}}}{1 + R^2_{\mathcal{B}\mathcal{A}}} + \ldots \\
\ldots - \left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot \frac{1 + R^2_{\mathcal{B}\mathcal{A}} - (R_{\mathcal{B}\mathcal{A}} + p_1)^2}{1 + R^2_{\mathcal{B}\mathcal{A}}} + \ldots \\
\ldots \frac{\left(1 + R^2_{\mathcal{B}\mathcal{A}}\right) \cdot \left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right)}{1 + R^2_{\mathcal{B}\mathcal{A}}}
\end{aligned}
\tag{B.4.19}
$$

i.e.

$$x_{\mathcal{E}} = \frac{x_{\mathcal{A}} + x_{\mathcal{B}} \cdot R_{\mathcal{B}\mathcal{A}}^2}{1 + R_{\mathcal{B}\mathcal{A}}^2} + \\ + h_{\mathcal{A}} \cdot \mathcal{F}_1 \cdot \frac{1 + R_{\mathcal{B}\mathcal{A}}^3 - (R_{\mathcal{B}\mathcal{A}} + p_1)^3 - \left[1 + R_{\mathcal{B}\mathcal{A}}^2 - (R_{\mathcal{B}\mathcal{A}} + p_1)^2\right]^{\frac{3}{2}}}{1 + R_{\mathcal{B}\mathcal{A}}^2} + \\ + \frac{\left(\frac{h_{\mathcal{D}}}{\tan(\alpha)} + \frac{h_{\mathcal{E}}}{\tan(\beta)}\right) \cdot (R_{\mathcal{B}\mathcal{A}} + p_1)^2}{1 + R_{\mathcal{B}\mathcal{A}}^2} \tag{B.4.20}$$

Now completely replacing $h_{\mathcal{D}}$ and $h_{\mathcal{E}}$ by their expressions (equations (B.3.6) and (B.3.11)) and using the reduced variables $r_p = R_{\mathcal{B}\mathcal{A}} + p_1$ and $r_1 = 1 + R_{\mathcal{B}\mathcal{A}}^2$:

$$x_{\mathcal{D}} = \frac{x_{\mathcal{A}} + x_{\mathcal{B}} \cdot R_{\mathcal{B}\mathcal{A}}^2 + h_{\mathcal{A}} \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2\right]^{\frac{3}{2}}\right) \cdot \mathcal{F}_1 - h_{\mathcal{A}} \cdot \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)}\right) \cdot \left[r_1 - r_p^2\right]}{r_1} \tag{B.4.21}$$

$$x_{\mathcal{E}} = \frac{x_{\mathcal{A}} + x_{\mathcal{B}} \cdot R_{\mathcal{B}\mathcal{A}}^2 + h_{\mathcal{A}} \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2\right]^{\frac{3}{2}}\right) \cdot \mathcal{F}_1 + h_{\mathcal{A}} \cdot \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)}\right) \cdot r_p^2}{r_1} \tag{B.4.22}$$

Consequently

$$\frac{x_{\mathcal{D}} - x_{\mathcal{B}}}{h_A} = \frac{\frac{x_{\mathcal{A}} - x_{\mathcal{B}}}{h_{\mathcal{A}}} + \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2\right]^{\frac{3}{2}}\right) \cdot \mathcal{F}_1 - \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)}\right) \cdot \left[r_1 - r_p^2\right]}{r_1} \tag{B.4.23}$$

and

$$\frac{x_{\mathcal{E}}-x_{\mathcal{B}}}{h_A}=\frac{\frac{x_{\mathcal{A}}-x_{\mathcal{B}}}{h_{\mathcal{A}}}+\left(1+R_{\mathcal{B}\mathcal{A}}^3-r_p^3-\left[r_1-r_p^2\right]^{\frac{3}{2}}\right)\cdot\mathcal{F}_1+\left(\frac{r_p}{\tan(\alpha)}+\frac{\sqrt{r_1-r_p^2}}{\tan(\beta)}\right)\cdot r_p^2}{r_1} \tag{B.4.24}$$

B.4.3 Summary of equations for height and position, coalescence and repulsion

We use the index $\mathcal{C}$ to designate the properties of a dune resulting from a merger and the indices $\mathcal{D}$ and $\mathcal{E}$ to designate the dunes that result from a splitting case (respectively upwind and downwind). $\mathcal{A}$ and $\mathcal{B}$ designate the upwind and downwind dunes before transformations.

Notations are the following:

$$\begin{cases} R_{\mathcal{B}\mathcal{A}} &= \dfrac{h_{\mathcal{B}}}{h_{\mathcal{A}}} \\ r_1 &= 1+R_{\mathcal{B}\mathcal{A}}^2 \\ r_p &= R_{\mathcal{B}\mathcal{A}}+p_1 \\ s &= \dfrac{x_{\mathcal{B}}}{x_{\mathcal{A}}} \\ t &= \dfrac{h_{\mathcal{A}}}{x_{\mathcal{A}}} \\ \mathscr{L} &= \dfrac{h_{\mathcal{D}}}{\tan(\alpha)}+\dfrac{h_{\mathcal{E}}}{\tan(\beta)} \\ \mathscr{F}_1 &= \dfrac{1}{3}\left(-\dfrac{1}{\tan(\alpha)}+\dfrac{1}{\tan(\beta)}\right) \end{cases} \tag{B.4.25}$$

We have:

$$\begin{cases} p_2 &= 1-\sqrt{r_1-r_p^2} \\ \dfrac{h_{\mathcal{C}}}{h_{\mathcal{A}}} &= \sqrt{r_1} \\ \dfrac{h_{\mathcal{D}}}{h_{\mathcal{A}}} &= r_p \\ \dfrac{h_{\mathcal{E}}}{h_{\mathcal{A}}} &= \sqrt{r_1-r_p^2} \end{cases} \tag{B.4.26}$$

$$\begin{cases} \dfrac{x_{\mathcal{C}}}{x_{\mathcal{A}}} &= \dfrac{1+R_{\mathcal{B}\mathcal{A}}^2 \cdot s}{r_1} + t \cdot \left(1 + R_{\mathcal{B}\mathcal{A}} - R_{\mathcal{B}\mathcal{A}} \cdot \dfrac{1+R_{\mathcal{B}\mathcal{A}}}{r_1} - \sqrt{r_1}\right) \cdot \mathcal{F}_1 \\ \dfrac{x_{\mathcal{D}}}{x_{\mathcal{A}}} &= \dfrac{1+s \cdot R_{\mathcal{B}\mathcal{A}}^2 + t \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2\right]^{\frac{3}{2}}\right) \cdot \mathcal{F}_1 - t \cdot \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)}\right) \cdot \left[r_1 - r_p^2\right]}{r_1} \\ \dfrac{x_{\mathcal{E}}}{x_{\mathcal{A}}} &= \dfrac{1+s \cdot R_{\mathcal{B}\mathcal{A}}^2 + t \cdot \left(1 + R_{\mathcal{B}\mathcal{A}}^3 - r_p^3 - \left[r_1 - r_p^2\right]^{\frac{3}{2}}\right) \cdot \mathcal{F}_1 + t \cdot \left(\frac{r_p}{\tan(\alpha)} + \frac{\sqrt{r_1 - r_p^2}}{\tan(\beta)}\right) \cdot r_p^2}{r_1} \end{cases} \tag{B.4.27}$$

Appendix C

Appendix to chapter 3

C.1 Details of the implementation

C.1.1 General structure of the code

The code first checks that the smoothing-window size is a multiple of 4, plus 1, and that the swipe length is an odd number. If these conditions are not satisfied, the code increases the parameter or parameters until both conditions are satisfied.

The centerline and both banks are subsequently smoothed using the filtering function, and the curvature function is run on the filtered centerline data. The width algorithm is then applied to the filtered datasets. The outputs desired by the user are organized and returned as variables.

C.1.2 Instructions for use

C.1.2.1 List of files and descriptions

The list of files included in our code is given in Table C.1.

C.1.2.2 How to run the code

Unzip all files into a single directory. Files 1 to 6 are the functions. Files 9 and 10 are the example datasets. Files 7 and 8 are the example Matlab© files (for use of the code) that you may want to open, just to look at how this was used.par

1	`Courbure.m`	Basic curvature code (recommended version for natural, noisy data), used by Width function with option `old=0`
2	`Curvature.m`	Basic curvature code (recommended version for simulated, clean and regularly spaced data), used by Width function with option `old=1`
3	`Filtering.m`	Code that filters/smoothes data, used by Width function automatically
4	`FiltFiltering.m`	Code that filters/smoothes data using Matlab©´s `filtfilt` function
5	`InterpTriangle.m`	Small interpolation code used by Width function (corresponds to equation (3.2.14))
6	`Width.m`	Main width function
7	`Ricker.m`	Ricker-wavelet creation code, used by Filtering function
8	`Trinity1952.m`	Code used to test Width code on Trinity river 1952 data
9	`Trinity2009.m`	Code used to test Width code on Trinity river 2009 data
10	`Trinity1952.txt`	Trinity river 1952 data
11	`Trinity2009.txt`	Trinity river 2009 data

Table C.1: List of files and description

The Width code is a function, so you do not have to open the file `Width.m` (nor any of files 1 to 6), except if you want or need to modify this file. Just run the function in the Matlab© command window with the selected parameters...

```
[ChanWidth, ChanWidth2, R, coord1, coord2, width1, width2, Ox, Oy,
xdir, ydir, coordfilt, bankfilt1, bankfilt2, mindist1, mindist2, maxival1,
maxival2]=Width(x, y, xbi1, ybi1, xbi2, ybi2, windowsize, theta, useold,
SwipeLength, filteringoff)
```

The inputs `x`, `y`, `xbi1`, `ybi1`, `xbi2`, `ybi2`, `windowsize`, `theta`, `useold` and `SwipeLength` are described in the next paragraph, while the outputs `ChanWidth`, `ChanWidth2`, `R`, `coord1`, `coord2`, `width1`, `width2`, `Ox`, `Oy`, `xdir`, `ydir`, `mindist1`, `mindist2`, `maxival1`, `maxival2` are described in the paragraph after the next one.

C.1.2.3 Table of necessary inputs

Inputs that are necessary to run the code are presented in Table C.2.

C.1.2.4 Table of possible outputs

Table C.3 contains the outputs that the user can obtain from the code.

A scatter plot of the outputs `ChanWidth2` versus `ChanWidth1` may indicate imperfections of the digitized dataset, such as an inadvertent switch of centerline data points and bank data points that puts the centerline outside of the banks, or areas where the width was not computed exactly along the centerline normal, etc. If the code is successful at processing the data, all points of the scatter plot should be on or very close to the $y = x$ line.

We did not use the output variables `Mindist*` and `Maxival*` for anything else than debugging.

C.1.2.5 Notes

Variable formats: Input and outputs are column vectors. We do not check for consistency (horizontal vectors, vectors of different sizes), and therefore the code will not work if you input a line vector. Best use is to load an input text file, as we

`x`:	contains the abscissae of centerline points
`y`:	contains the ordinates of centerline points
`xbi1`:	contains the abscissae of points on bank 1
`ybi1`:	contains the ordinates of points on bank 1
`xbi2`:	contains the abscissae of points on bank 2
`ybi2`:	contains the ordinates of points on bank 2
`windowsize`:	is the size of the smoothing window, and depends on how tight the bends are. A larger window size reduces scattering but increases the error in tight bends by cutting through these bends. The window size must be a (multiple of 4)+1. We use a window size of 21 points for the Trinity river, and of 41 points for Mississippi river.
`theta`:	is the proportion of Ricker (theta) vs Chebyshev (1-theta) window for filtering process. The Chebyshev window smoothes better but cuts more corners. We only used a Ricker window for the Trinity river (`theta=1`), and used a value of `theta=0.6` for the Mississippi river.
`useold`:	determines which curvature code is applied. `useold=1` applies equations 4 and 5, whereas `useold=0` applies equation 7. We used `useold=0` for both the Trinity and Mississippi rivers. `useold=0` handles digitization error (irregular distance between points) better than `useold=1` but vertical tangents may cause inaccuracies.
`swipelength`:	is the length of tests for normal alignment around the minimum distance point (see section 3.2.3). You should choose a number small enough to avoid reaching on the other side of a bend, but large enough to find the centerline-normal when it is far from the minimum-distance point. The value has to be an odd number (X points on the left of the minimum-distance point, minimum-distance point, X points on the right of the minimum distance point$= 2 \times X + 1$). We used $11 = 2 \times 5 + 1$ points for the Trinity river. That length is automatically doubled, in a second pass, when the code cannot find a width value in the first pass.
`filteringoff`:	bypasses the filtering process (only raw data are used).

Table C.2: Description of function inputs

ChanWidth:	contains the widths W_C of channel as defined in section 3.2.3
ChanWidth2:	contains the widths W_D of channel as defined in section 3.2.3
R:	contains the radii of curvature of the centerline
Coord1:	contains the coordinates of the intersections of bank 1 and the centerline-normal at each centerline point. This variable is a two-column vector (one column for X, one for Y).
Coord2:	contains the coordinates of the intersections of bank 2 and the centerline-normal at each centerline point. This variable is a two-column vector (one column for X, one for Y).
Width1:	contains the distances from bank 1 to the centerline
Width2:	contains the distances from bank 2 to the centerline
Ox:	are the abscissae of the centers of curvature of the centerline
Oy:	are the ordinates of the centers of curvature of the centerline
Xdir:	contains the X-coordinates of the normalized centerline-normal vectors
Ydir:	contains the Y-coordinates of the normalized centerline-normal vectors
coordfilt:	contains the filtered centerline coordinates
bankfilt1:	contains the filtered coordinates of bank 1
bankfilt2:	contains the filtered coordinates of bank 2
Mindist1:	is the list of indices of points on bank 1 that are at a minimum distance of centerline points
Mindist2:	is the list of indices of points on bank 2 that are at a minimum distance of centerline points
Maxival1:	represents the number of points on bank 1 between the minimum-distance point (`Mindist1`) and the point P that maximizes the dot product between the centerline-normal vector and vector centerline point-P
Maxival2:	represents the number of points on bank 2 between the minimum-distance point (`Mindist2`) and the point P that maximizes the dot product between the centerline-normal vector and vector centerline point-P

Table C.3: Description of function outputs

did in the examples.

Output variables: You do not have to specify every output variable, but you have to specify every variable before the one you want. For instance if you do not care about `coord1`, `coord2`, `width1`, `width2`, `Ox`, `Oy`, `xdir`, `ydir`, etc., but want the width(s) and radius of curvature, you can just run

```
[ChanWidth,ChanWidth2,R]=Width2(x,y,xbi1,ybi1,xbi2,ybi2,
windowsize,theta,useold,SwipeLength)
```

However, you do have to specify every input variable except the last four (`useold`, `windowsize`, `theta`, `swipelength`). If you do not specify the last four variables, then the program defaults on the values we used for the Trinity River

Example files: Look at the `Trinity1952.m` and `Trinity2009.m` files to see how the data are loaded, how the code is run and how to plot left and right banks, centerline, and width-calculation lines. If you would like to plot point numbers (every 5^{th} point) next to the centerline points (plotting takes longer but it makes it easier to match features between plot and data), uncomment the lines reading:

```
if mod(j,5)==0
text(CenterLineX(j),CenterLineY(j),num2str(j));
end
```

Filtering function: We allow the user to use Matlab©'s `filtfilt` function to filter the data instead of our own filtering function by renaming the `Filtering.m` file to something else, and then renaming the `FiltFiltering.m` file to `Filtering.m`.

No value points: The code checks for "no-value" points (Matlab©'s `NaN`). Indeed, if an input vector contains such points, the filtered vector contains many more no-value points:

- when filtering through the `Filtering.m` function, one `NaN` point at index $i_{\texttt{NaN}}$ results in L (total length of filtering window) `NaN` points centered on $i_{\texttt{NaN}}$; if $i_{\texttt{NaN}} < \frac{L+1}{2}$ (respectively $i_{\texttt{NaN}} > N - \frac{L+1}{2}$), all points from 1 to $i_{\texttt{NaN}} + \frac{L-1}{2}$ (respectively from $i_{\texttt{NaN}} - \frac{L-1}{2}$ to N) will be transformed to `NaN` by the filtering process.

- when filtering through the `FiltFiltering.m` function, one single `NaN` value in the input results in a `NaN` everywhere.

Therefore, to avoid empty result datasets, the filtering code ignores points where either the abscissa or the ordinate is undefined. Such points are left unfiltered, and the user is warned by a pop-up dialog. This means that the filtering window will effectively extend one point more over each `NaN` value. If, for instance, you are filtering the data with an 11-point window (five points on each side of the middle-point), and point 20 is undefined, then the filtering for points 15, 16, 19, 21, 24 and 25 will respectively extend from points 10 to 26, 11 to 27, 14 to 25, 15 to 26, 18

to 29 and 19 to 30; point 20 will simply be ignored and stay as `NaN` in the filtered vector.

Special issues: Please note that this code does not use a match between point indices in the centerline and point indices in the banks, which allows a large difference in the total number of points between the centerline and the banks. Instead, the center-point for the "swipe" is the point in the given bank that is the closest to the centerline point we consider. Therefore, problems may happen if the banks are shaped such that the closest point to the centerline is farther away than $\frac{\text{Swipe Length}-1)}{2}$ from the point on the intersection of the bank and the normal to the centerline. In such a case, a sequence of consecutive points does not initially get a value, but a second pass tries to correct for this problem. Another issue is when the centerline and the banks do not run parallel to each other but rather perpendicular to each other, in which case we cannot really define a width the same way as elsewhere, and such points are left with `NaN` values.

Computer requirements: Running this code requires a licensed copy of Matlab©, with possibly a few toolboxes such as the statistics toolbox; Matlab© 2011a was used to test the code, but we suppose that earlier versions of Matlab© 7.x can also run the code. Student versions are not advised, as the memory available to the application is restricted. All recent computer processors ($>$ 2007) should be able to process the code. However, you may experience "out-of-memory" errors depending on the amount of RAM of your computer. Indeed, the code is sped up through

the use of vectorized functions, as Matlab© is much slower when running loops; for instance, matrices as large as $M \times N$ are used, where M is the number of points of the river centerline and N the maximum number of points of the river banks. If your computer does not have enough memory to run the code as-is, vectorized parts of the code may be transformed into loops at the expense of speed.

Licensing: The author retains full ownership of the code. You may run the code freely and share it, as long as authorship information is provided. The author is aware that improvements of this code are possible and encourages you to improve the code; however, for the general benefits of the scientific community, we require that such modifications be made public. We also wish for a graphical interface to be created to use the code, as we know that long lines of parameters, although useful to customize the way to run the code, are impractical but a necessary evil when there is no graphical interface.

C.1.3 Fast detection of local extrema

We detail in this section our method to detect the local extrema of a function without resorting to complex programming. We apply this method on each geometrical variable (radius of curvature, bottom length and bottom slope, total depth, channel width) even though we describe it in this section for only one variable that we call $V(x)$, where x is the streamwise distance along the channel centerline. The steps we apply are the following:

1. We apply a low-pass filter (a Chebyshev window) on $V(x)$ to obtain a low-

frequency version of $V(x)$ that we call $\overline{V(x)}$; the filter lengths we used were $103 = \frac{N}{10}$ points for S_2^s and R, and $52 \simeq \frac{N}{20}$ for the other geometrical parameters; We add the overbar sign in the figure captions to designate the low-frequency version of a specific variable

2. We assume that there is an extremum of $V(x)$ between each intersection of $V(x)$ and $\overline{V(x)}$. These intersections have positions x_{inter} on the channel centerline as given by equation (C.1.1)

$$\{x_{inter}\} = \left\{x \;/\; V(x) - \overline{V(x)} = 0\right\} \tag{C.1.1}$$

3. The low-frequency filter creates a moving average $\overline{V(x)}$ that may not cross the original function $V(x)$ each time there is a change in the sign of the rate of change of $V(x)$ – a problem that happens mostly with rapid variations of $V(x)$ outside of local trends (maxima much lower than neighboring maxima, minima much higher than neighboring minima), where a higher-frequency moving-average filter would be required –; therefore we slightly adjust equation (C.1.1) by introducing a coefficient α that we choose by trial and error so it maximizes the number of intersections of $\overline{V(x)}$ with $V(x)$. Equation(C.1.1) becomes equation (C.1.2):

$$\{x_{inter}\} = \left\{x \;/\; V(x) - \alpha \cdot \overline{V(x)} = 0\right\} \tag{C.1.2}$$

Even with these corrections, this technique is not perfect and some variations are missed; However, it allows us to find most extrema with only a few lines

of code, very little processing power and much faster than conventional gradient methods. The α coefficient is not the same for all variables. For instance, we used $\alpha = \frac{1}{0.8}$ when we looked for minima of $\left|S_2^s\right|$, $\alpha = 0.95$ for the determination of the maxima of $|R|$, $\alpha = 0.8$ when looking for maxima of $\left|S_2^s\right|$ and $\alpha = \frac{1}{0.95}$ for the minima of $|R|$.

4. We then look for global extrema between the positions $x_{inter}(i)$ and $x_{inter}(i+1)$, $i \in [\![1; N_{inter}]\!]$ (where N_{inter} is the cardinal of the whole set of curvilinear abscissæ x_{inter} as defined by equation (C.1.2)) of the intersections of $V(x)$ with $\alpha \cdot \overline{V(x)}$, and assume that the global extremum in each $[x_{inter}(i); x_{inter}(i+1)]$ interval is a local extremum of the function $V(x)$.

We provide an example of extremum detection (maxima of $|R|$ and minima of $\left|S_2^s\right|$) in figure C.1 and we magnify the $[130; 150]$ km area in Figure C.2.

C.2 Curvature of a curve described by Von Schelling's equation

Using the slope of the tangent to the curve given by equation (3.1.1), we obtain a tangent vector $\overrightarrow{T(x)} = \begin{bmatrix} \Phi_X(x) \\ \Phi_Y(x) \end{bmatrix}$ so that $\tan[\phi(x)] = \frac{T_Y(x)}{T_X(x)}$:

$$\overrightarrow{\Phi(x)} = \begin{bmatrix} \cos[\phi(x)] \\ \sin[\phi(x)] \end{bmatrix} = \begin{bmatrix} \cos\left[\omega \cdot \sin\left(2 \cdot \pi \cdot \frac{x}{M}\right)\right] \\ \sin\left[\omega \cdot \sin\left(2 \cdot \pi \cdot \frac{x}{M}\right)\right] \end{bmatrix} \tag{C.2.1}$$

The vector is already normalized. The normal vector $\overrightarrow{N(x)}$ is

$$\overrightarrow{N(x)} = \begin{bmatrix} -T_Y(x) \\ T_X(x) \end{bmatrix} = \begin{bmatrix} -\sin[\phi(x)] \\ \cos[\phi(x)] \end{bmatrix} \tag{C.2.2}$$

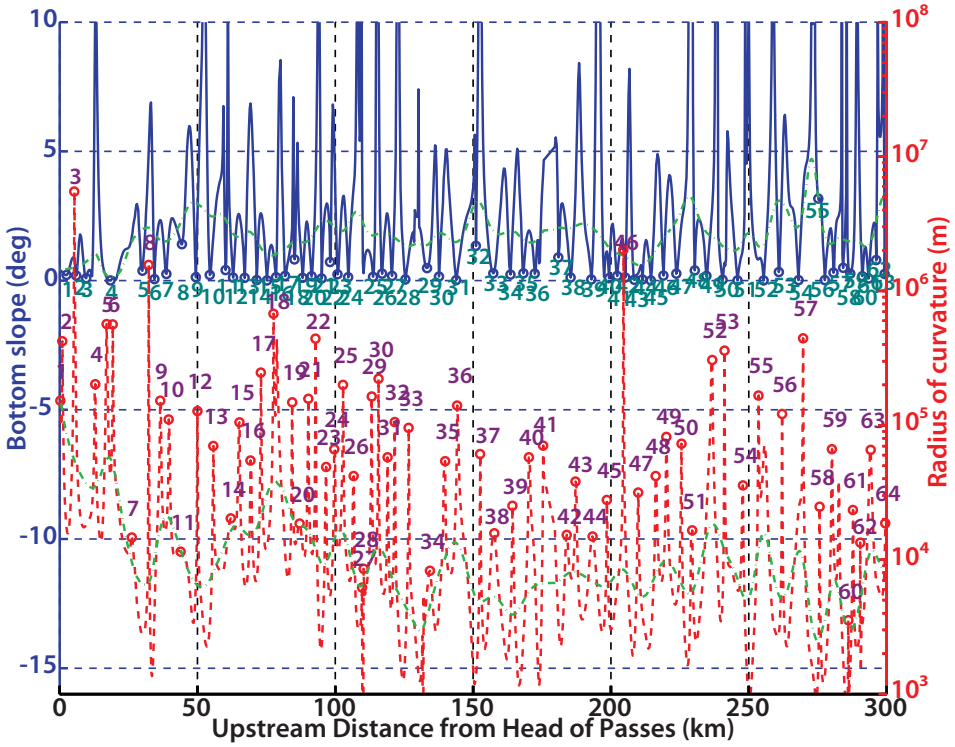


Figure C.1: Bottom slope $\left|S_2^s\right|$ (continuous blue line, linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes. The two green dot-dashed lines are $\overline{\left|S_2^s\right|}$ and $\overline{R}$. Detected local maxima of $\left|S_2^s\right|$ and $|R|$ are displayed with (respectively blue and red) dots and numbered.

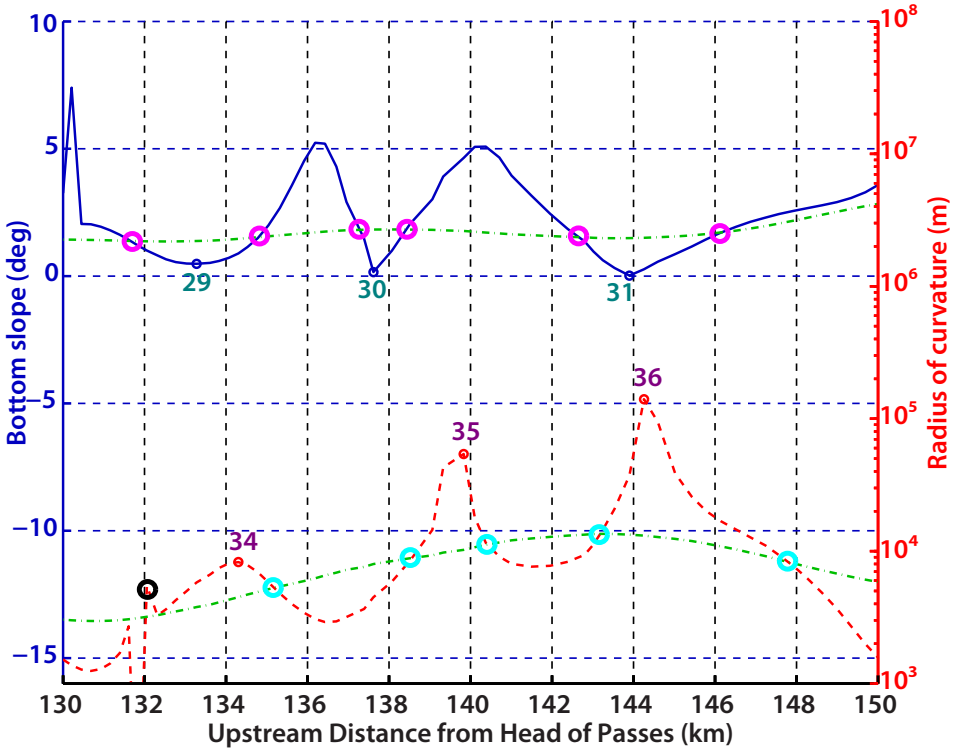


Figure C.2: Bottom slope $\left|S_2^s\right|$ (continuous blue line, linear scale) and unsigned radius of curvature $|R|$ (dashed red line, logarithmic scale) as functions of upstream distance from Head of Passes, zoom over $[130;150]$ km area. The two green dot-dashed lines are $\overline{\left|S_2^s\right|}$ and $\overline{R}$. Detected local maxima of $\left|S_2^s\right|$ and $|R|$ are displayed with (respectively blue and red) dots and numbered. The intersections of $\alpha_R \cdot \overline{R}$ with $|R|$, and of $\alpha_S \cdot \overline{\left|S_2^s\right|}$ and $\left|S_2^s\right|$ are respectively displayed with cyan and pink dots. Local minima of $\left|S_2^s\right|$ are located between two pink dots, and local maxima of $|R|$ are located between two cyan dots. A black dot shows a local extremum that the algorithm could not detect.

By definition, the curvature $|\gamma(x)|$ is the norm of the derivative of $\overrightarrow{T(x)}$ with respect to x:

$$\gamma(x) = \left\| \frac{\mathrm{d}\overrightarrow{T(x)}}{\mathrm{d}x} \right\| \tag{C.2.3}$$

and the signed curvature $\gamma(x)$ is such that

$$\frac{\mathrm{d}\overrightarrow{T(x)}}{\mathrm{d}x} = \gamma(x) \cdot \overrightarrow{N(x)} \tag{C.2.4}$$

so let's compute this derivative of $\overrightarrow{T(x)}$:

$$\begin{aligned} \frac{\mathrm{d}\overrightarrow{T(x)}}{\mathrm{d}x} &= \frac{\mathrm{d}\phi(x)}{\mathrm{d}x} \cdot \begin{bmatrix} -\sin[\phi(x)] \\ \cos[\phi(x)] \end{bmatrix} \\ &= \frac{\mathrm{d}\phi(x)}{\mathrm{d}x} \cdot \overrightarrow{N(x)} \end{aligned} \tag{C.2.5}$$

hence

$$\gamma(x) = \frac{\mathrm{d}\phi(x)}{\mathrm{d}x} = \frac{2 \cdot \pi \cdot \omega}{M} \cdot \cos\left(2 \cdot \pi \cdot \frac{x}{M}\right) \tag{C.2.6}$$

and

$$\boxed{R(x) = \frac{M}{2 \cdot \pi \cdot \omega} \cdot \frac{1}{\cos\left(2 \cdot \pi \cdot \frac{x}{M}\right)}} \tag{C.2.7}$$

In terms of $\phi(x)$,

$$R(x) = \frac{1}{\frac{\mathrm{d}\phi(x)}{\mathrm{d}x}} \tag{C.2.8}$$

When $\frac{x}{M} \cong 0\left[\frac{1}{2}\right]$, $\cos\left(2\cdot\pi\cdot\frac{x}{M}\right) = \pm 1$, which obviously gives us the minimum of $|R|$:

$$R_{\min} = R\left(\frac{x}{M} = 0\right) = R\left(\frac{x}{M} = \frac{1}{2}\right) = \ldots = \frac{M}{2\cdot\pi\cdot\omega} \tag{C.2.9}$$

When ω is expressed in degrees, which is often the case in geological publications, $R_{\min}\cdot\frac{\omega_{\deg}}{M} = \frac{90}{\pi^2} \simeq 9.119$.

Finding a Cartesian equation of the curve is a real challenge, if possible at all. However, we can indicate the coordinates of the segments/vectors $\overrightarrow{R(x)}$ that join a point on the curve with its associated center of curvature, given that $\overrightarrow{R(x)} = R(x)\cdot\overrightarrow{N(x)}$:

$$\overrightarrow{R(x)} = \frac{1}{\frac{\mathrm{d}\phi(x)}{\mathrm{d}x}}\cdot\begin{bmatrix}-\sin[\phi(x)]\\ \cos[\phi(x)]\end{bmatrix} = \frac{M}{2\cdot\pi\cdot\omega}\cdot\frac{1}{\cos\left(2\cdot\pi\cdot\frac{x}{M}\right)}\cdot\begin{bmatrix}-\sin\left[\omega\cdot\sin\left(2\cdot\pi\cdot\frac{x}{M}\right)\right]\\ \cos\left[\omega\cdot\sin\left(2\cdot\pi\cdot\frac{x}{M}\right)\right]\end{bmatrix} \tag{C.2.10}$$

C.3 Amplitude of a period of a curve described by Von Schelling's equation

If X and Y are coordinates of a "von Schelling curve" in an orthonomal basis, then for $0 < x < \frac{M}{2}$

$$\frac{\mathrm{d}X}{\mathrm{d}x} = \cos[\phi(x)] \tag{C.3.1}$$

and

$$\frac{\mathrm{d}Y}{\mathrm{d}x} = \sin\left[\phi\left(x\right)\right] \tag{C.3.2}$$

i.e.

$$X\left(\frac{M}{2}\right) = \int_0^{\frac{M}{2}} \cos\left[\omega \cdot \sin\left(\frac{2\cdot\pi}{M}\cdot x\right)\right] \mathrm{d}x \tag{C.3.3}$$

and

$$Y\left(\frac{M}{2}\right) = \int_0^{\frac{M}{2}} \sin\left[\omega \cdot \sin\left(\frac{2\cdot\pi}{M}\cdot x\right)\right] \mathrm{d}x \tag{C.3.4}$$

We may get an expression for $X\left(\frac{M}{2}\right)$ by using a variable change, $\delta = \frac{2\cdot\pi}{M}\cdot x$,

$$\begin{aligned} X\left(\frac{M}{2}\right) &= \frac{M}{2\cdot\pi}\cdot\int_0^{\pi} \cos\left[\omega\cdot\sin\left(\delta\right)\right]\mathrm{d}\delta \\ &= \frac{M}{2}\cdot\frac{1}{\pi}\cdot\int_0^{\pi} \cos\left[0\cdot\delta - \omega\cdot\sin\left(\delta\right)\right]\mathrm{d}\delta \end{aligned} \tag{C.3.5}$$

We recognize here the definition of a Bessel $\mathbf{J}$ function of order 0:

$$X\left(\frac{M}{2}\right) = \frac{M}{2}\cdot\mathbf{J}_0\left(\omega\right) \tag{C.3.6}$$

The variable change is slightly different for $Y\left(\frac{M}{2}\right)$, as we use $\eta = \frac{\pi}{2} - \frac{2\cdot\pi}{M}\cdot x$,

then $u = -\eta$:

$$\begin{aligned} Y\left(\frac{M}{2}\right) &= \frac{M}{2\cdot\pi}\cdot\int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}}\sin\left[\omega\cdot\cos\left(\eta\right)\right]\mathrm{d}\eta \\ &= \frac{M}{2\cdot\pi}\cdot\int_{-\frac{\pi}{2}}^{0}\sin\left[\omega\cdot\cos\left(\eta\right)\right]\mathrm{d}\eta + \frac{M}{2\cdot\pi}\cdot\int_{0}^{\frac{\pi}{2}}\sin\left[\omega\cdot\cos\left(\eta\right)\right]\mathrm{d}\eta \\ &= \frac{M}{2\cdot\pi}\cdot\int_{0}^{\frac{\pi}{2}}\sin\left[\omega\cdot\cos\left(u\right)\right]\mathrm{d}u + \frac{M}{2\cdot\pi}\cdot\int_{0}^{\frac{\pi}{2}}\sin\left[\omega\cdot\cos\left(\eta\right)\right]\mathrm{d}\eta \end{aligned} \tag{C.3.7}$$

Given that η and u are dummy (bound) variables, and that $\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$ (gamma function),

$$\begin{aligned} Y\left(\frac{M}{2}\right) &= \frac{M}{\pi}\cdot\int_{0}^{+\frac{\pi}{2}}\sin\left[\omega\cdot\cos\left(\eta\right)\right]\mathrm{d}\eta \\ &= \frac{M}{2}\cdot\frac{2\cdot\left(\frac{\omega}{2}\right)^{0}}{\sqrt{\pi}\cdot\Gamma\left(0+\frac{1}{2}\right)}\int_{0}^{\frac{\pi}{2}}\cdot\sin\left[\omega\cdot\cos\left(\eta\right)\right]\cdot\left[\sin\left(\eta\right)\right]^{2\times 0}\mathrm{d}\eta \end{aligned} \tag{C.3.8}$$

We recognize here the definition of a Struve $\mathbf{H}$ function of order 0:

$$Y\left(\frac{M}{2}\right) = \frac{M}{2}\cdot\mathbf{H}_0\left(\omega\right) \tag{C.3.9}$$

Because of the symmetry properties of the curve, the amplitude λ of a period in X is twice the value obtained for $x = \frac{M}{2}$, while the amplitude a in Y is the value obtained for $x = \frac{M}{2}$:

$$\begin{cases} \lambda\left(M,\omega\right) = M\cdot\mathbf{J}_0\left(\omega\right) \\ a\left(M,\omega\right) = \dfrac{M}{2}\cdot\mathbf{H}_0\left(\omega\right) \end{cases} \tag{C.3.10}$$

Figure C.3 shows $\frac{a(M,\omega)}{M} = \frac{\mathbf{H}_0(\omega)}{2}$ as a function of $\frac{\lambda(M,\omega)}{M} = \mathbf{J}_0(\omega)$. The angle ω is increasing from the first point $\omega = 0$ at $(\lambda, a) = (1,0)$ to $\omega = 122^o$.

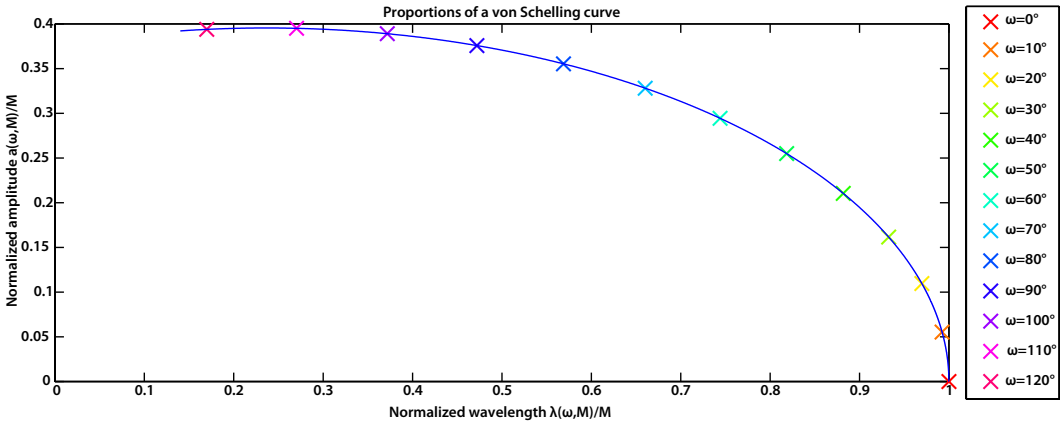


Figure C.3: Normalized wavelength vs. normalized amplitude for all physically possible values of ω

While it is not possible to quickly compute by hand an accurate value of the Bessel **J** and Struve **H** functions, these functions are common enough to have been implemented in current software or to be available in tables.

Additionally, we may want to know what angle $\omega_{lim} > \frac{\pi}{2}$ is needed for two consecutive bends (half-meanders) to touch each other (maximum value for a realistic river model). Two consecutive bends will touch, for a large enough value of ω, at points $x_\perp$ that exhibit a vertical tangent. In the first bend, the first of these points is such that:

$$x_\perp(M,\omega) = \frac{M}{2\cdot\pi}\cdot\arcsin\left(\frac{\pi}{2\cdot\omega}\right) \tag{C.3.11}$$

Due to the symmetry of a meander, the bends will touch if and only if $2\cdot$

$X_\perp(M,\omega) = \lambda(M,\omega_{\lim})$, where $X_\perp$ is the Cartesian abscissa corresponding to $x_\perp$:

$$X_\perp = \int_0^{\frac{M}{2\cdot\pi}\cdot\arcsin\left(\frac{\pi}{2\cdot\omega}\right)} \cos\left[\omega\cdot\sin\left(\frac{2\cdot\pi}{M}\cdot x\right)\right] \mathrm{d}x \tag{C.3.12}$$

With $\phi = \omega\cdot\sin\left(\frac{2\cdot\pi}{M}\cdot x\right)$, $\frac{M}{2\cdot\pi}\cdot\arcsin\left(\frac{\phi}{\omega}\right) = x$, $\mathrm{d}x = \frac{M}{2\cdot\pi}\cdot\frac{\mathrm{d}\phi}{\sqrt{\omega^2-\phi^2}}$

$$X_\perp = \frac{M}{2\cdot\pi\cdot\omega}\cdot\int_0^{\frac{\pi}{2}} \frac{\cos(\phi)}{\sqrt{1-\left(\frac{\phi}{\omega}\right)^2}}\mathrm{d}\phi \tag{C.3.13}$$

Therefore, if we suppose a constant river width B, $\omega_{\lim}$ is solution of the equation:

$$2\cdot(X_\perp + B) = \lambda(\omega_{\lim}, M) \tag{C.3.14}$$

This means that the condition of possible existence such a river is:

$$\frac{B}{M} \leq \frac{1}{2}\cdot\mathbf{J}_0(\omega) - \frac{1}{2\cdot\pi\cdot\omega}\cdot\int_0^{\frac{\pi}{2}} \frac{\cos(\phi)}{\sqrt{1-\left(\frac{\phi}{\omega}\right)^2}}\mathrm{d}\phi \tag{C.3.15}$$

or

$$\frac{B}{\lambda(\omega,M)} \leq \frac{1}{2} - \frac{1}{2\cdot\pi\cdot\omega\cdot\mathbf{J}_0(\omega)}\cdot\int_0^{\frac{\pi}{2}} \frac{\cos(\phi)}{\sqrt{1-\left(\frac{\phi}{\omega}\right)^2}}\mathrm{d}\phi \tag{C.3.16}$$

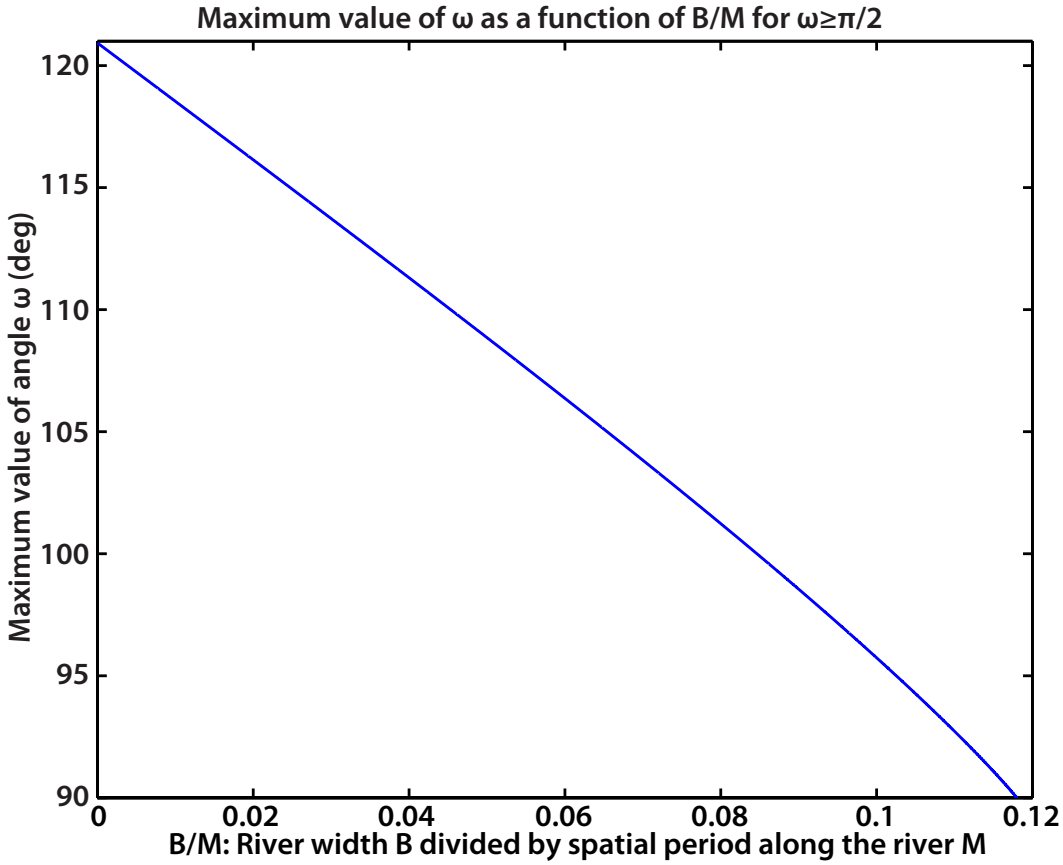


Figure C.4: Maximal value of angle ω as a function of B/M (river width divided by spatial period along the river), as given by (in)equation C.3.15.

Equation (C.3.15) is plotted in Figure C.4 and equation (C.3.16) is plotted in Figure C.5. We hope that such figures will be useful when designing laboratory experiments, as they help determining the maximum angle ω when B is known, or the maximum value of B when ω is known.

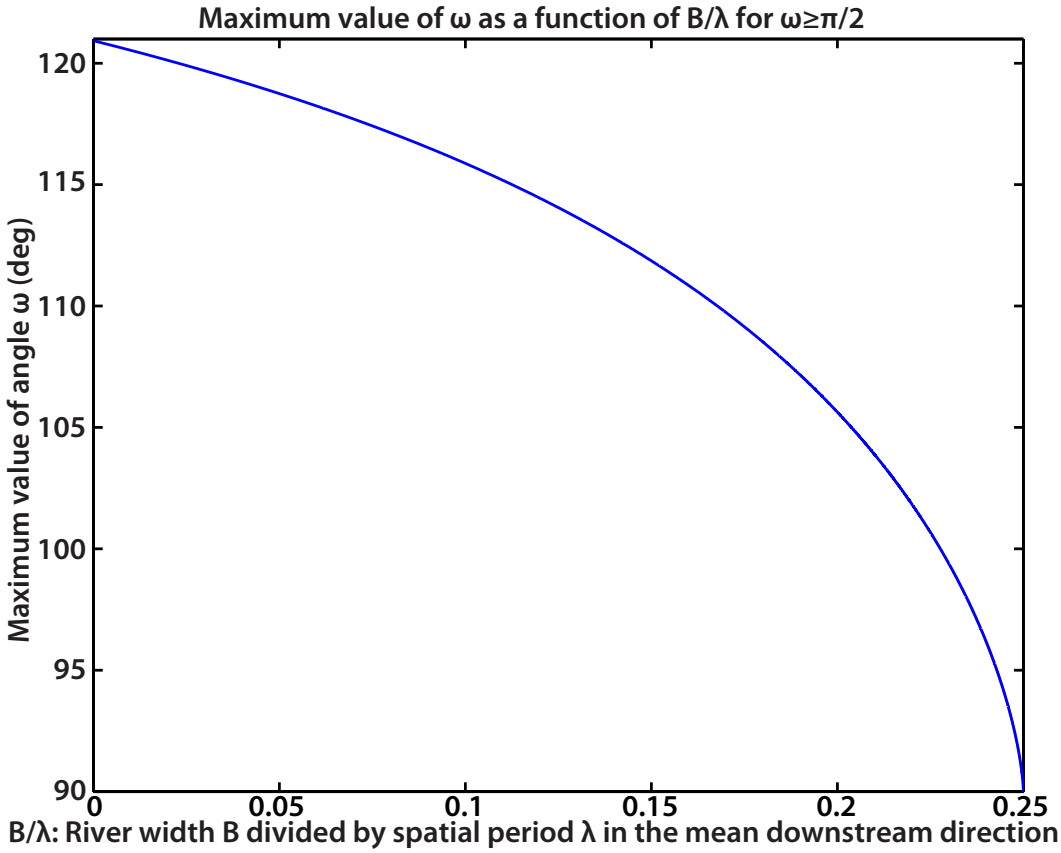


Figure C.5: Maximal value of angle ω as a function of B/λ (river width divided by spatial period along the mean downstream direction), as given by (in)equation C.3.16.

C.4 Additional code

We have written auxiliary functions to simplify the data processing involved with large datasets such as those we used. This code is provided for user convenience but has not been used to produce the results detailed in this article (often because this code was written after the result analysis which proved to contain many repetitive tasks). The particularity of this code is that it is not commented nor ex-

tensively tested and therefore may not always be efficient, although we believe it performs well in the majority of cases.

C.4.1 Automatic correlation of extrema

One of the most critical steps in the data analysis was the correlation between extrema of two variables: the unsigned radius of curvature $|R|$ on one hand and a geometrical parameter $|\mathcal{G}|$ on the other. Because of the varying lag between extrema of the two variables, direct correlation by cross-plot does not yield any result, and manual correlation is a very time-consuming, repetitve task; hence the need to automate the extremum-matching process described in section 3.3.3 on page 157; we use in this paragraph the same notations as in section 3.3.3. The `PickEx.m` function performs this process with additional enhancements:

1. When the user does not specify any value for the length of the low-pass filtering window, this length automatically defaults to 5% of the total variable length (rounded to the nearest integer);

2. The user may choose fixed values for the $\alpha_{...}$ and $\beta_{...}$ coefficients, but may also request the code to search for values of these coefficients that maximize the number of extrema found. In this case, the user specifies the center values α_0 and β_0 and the code computes the number of extrema for each value $\alpha = \alpha_0 + \delta$, $\beta = \beta_0 + \delta$ then chooses the α and β which produces the maximum number of extrema. δ takes all values between -0.25 and 0.25 by intervals of 0.01: $\delta \in \frac{[\![-25;+25]\!]}{100}$;

3. Correlations between extrema of one variable and extrema of the other minimize the distance between matched extrema;

4. It is possible that one single point of a variable is matched to different points of the other variable; In this case, only the match with the shortest distance between extrema will be retained;

5. The used may choose between forcing a negative correlation (minima of variable 1 matched with maxima of variable 2 and maxima of variable 1 matched with minima of variable 2), forcing a positive correlation (minima of variable 1 matched with minima of variable 2 and maxima of variable 1 matched with maxima of variable 2) or letting the code decide. In this case, the code computes matching points for both negative and positive correlations, then chooses which one is best based on the correlation coefficient between both sets of matching extrema or on the average minimum lag between matching extrema; We advise the user to take the correlation coefficient option, as the corresponding results seem to match hand-picked type of correlations in every case.

6. Once the correlation has been computed or found, the equations for the least-squares (minimized vertical distance fit or minimized orthogonal distance fit) fits are displayed;

7. The user needs to tell the code which scaling to use: linear or logarithmic. We advise a logarithmic scale for radius of curvature, and to try linear of

logarithmic for other variables, then decide of which scale to use after a look at resulting cross-plots.

This code may also be used to track migration patterns of rivers: minima of radius of curvature will be located at bend apices and maxima at inflexion points. By running this code on time-lapse datasets, ater discarding results other than extrema of the radius of curvature, the user may expect to be able to track the movement of bend apices and inflexion points in time, and possibly of whole meanders after matching intermediary points proportionally using the distance between an inflexion point and the next bend. Command-line use is as follows:

```
[d11,ex11,d12,ex12,d21,ex21,d22,ex22,v11,v12,v21,v22,sl,ic,slo,
ico,vardir,a1b,a2b,b1b,b2b]=PickEx(Distance1,var1,Distance2,var2,
BMin,BMax,linlog1,linlog2,invornot,a1,a2,b1,b2,findbest,WL1,WL2,
cororlag)
```

One lesser-known Matlab© trick is the use of the tilde (~) as a replacement for unwanted outputs. For instance, if you are just interested in the value of output parameter `a1b`, you may use the function with the following output:

```
[~,~,~,~,~,~,~,~,~,~,~,~,~,~,~,~,~,a1b,~,~,~]
```

However, if you plan on using the other pieces of auxilary code we have written, we advise to run the code with full outputs.

C.4.1.1 Table of inputs

Inputs of function `PickEx` are listed in Table C.4.

Name	Description	Length
`Distance1`	Distance along the channel associated with variable 1; needs to be monotonously increasing	L_1
`var1`	Variable 1: First variable to be correlated	L_1
`Distance2`	Distance along the channel associated with variable 2; needs to be monotonously increasing and unit has to be the same as that of Variable 1	L_2
`var2`	Variable 2: Second variable to be correlated	L_2
`BMin` (for `BoundaryMin`)	Distance along the channel where correlations should begin; set this parameter to any negative number to begin at the first point	1
`BMax` (for `BoundaryMax`)	Distance along the channel where correlations should end; to end correlations at the last point, set this parameter to a negative number or to any number smaller than `BoundaryMin`	1
`linlog1`	Scale of Variable 1: 0 for linear and 1 for exponential (i.e. requires plotting in logarithmic scale); Radius of curvature usually requires a logarithmic scale	1

`linlog2`	Scale of Variable 2: 0 for linear and 1 for exponential (i.e. requires plotting in logarithmic scale)	1
`invornot`	Type of correlation: $+1$ for positive correlation (minima of Variable 1 correspond to minima of Variable 2, maxima to maxima), -1 for negative correlation (minima of Variable 1 correspond to maxima of Variable 2, maxima to minima), 0 for automatic determination	1
`a1` (for `alpha1`)	Coefficient multiplying low-pass filtered-variable 1, which is then subtracted from Variable 1 to help determine minima of Variable 1	1
`a2` (`alpha2`)	Coefficient multiplying low-pass filtered-variable 1, which is then subtracted from Variable 1 to help determine maxima of Variable 1	1
`b1` (`beta1`)	Coefficient multiplying low-pass filtered-variable 2, which is then subtracted from Variable 1 to help determine minima of Variable 2	1
`b1` (`beta2`)	Coefficient multiplying low-pass filtered-variable 2, which is then subtracted from Variable 1 to help determine maxima of Variable 2	1

`findbest`	Automatic maximization of number of extrema: 0 keeps user-input of $\alpha_{1,2}$ and $\beta_{1,2}$, while 1 searches for values of $\alpha_{1,2}$ and $\beta_{1,2}$ that maximize the number of extrema found using the process described in bullet 2 of section C.4.1	1
`WL1` (for `WLength1`)	Length of low-pass filter window for Variable 1; defaults to $\left\lfloor \frac{L_1}{20} \right\rceil$ if input is ommitted	1
`WL2` (for `WLength2`)	Length of low-pass filter window for Variable 2; defaults to $\left\lfloor \frac{L_2}{20} \right\rceil$ if input is ommitted	1
`cororlag`	Method used to choose type of correlation: 0 for the minimum average lag, 1 for the maximum correlation coefficient; Default is 1	1

Table C.4: Description of `PickEx.m` function inputs

C.4.1.2 Table of outputs

Outputs of function `PickEx` are listed in Table C.5.

Name	Description	length
`d11`	Distances to the minima of variable 1 whose indices are given in `ex11`	L_3
`ex11`	Indices of minima of variable 1 matched with (minima/maxima) of variable 2	L_3

`d12`	Distances to the extrema of variable 2 whose indices are given in `ex12`	L_3
`ex12`	Indices of (minima/maxima) of variable 2 matched with minima of variable 1	L_3
`d21`	Distances to the maxima of variable 1 whose indices are given in `ex21`	L_4
`ex21`	Indices of maxima of variable 1 matched with (maxima/minima) of variable 2	L_4
`d22`	Distances to the extrema of variable 2 whose indices are given in `ex22`	L_4
`ex22`	Indices of (maxima/minima) of variable 2 matched with maxima of variable 1	L_4
`v11`	Values of minima of Variable 1 whose indices are given in `ex11`	L_3
`v12`	Values of extrema of Variable 2 whose indices are given in `ex12`	L_3
`v21`	Values of maxima of Variable 1 whose indices are given in `ex21`	L_4
`v22`	Values of extrema of Variable 2 whose indices are given in `ex22`	L_4
`sl` (for `slope`)	Slope of vertical least-square fit for all matched extrema	1
`ic` (for `intercept`)	Intercept (ordinate at origin) of vertical least-square fit for all matched extrema	1

`slo` (for `slopeortho`)	Slope of orthogonal least-square fit for all matched extrema	1
`ico` (for `interceptortho`)	Intercept (ordinate at origin) of orthogonal least-square fit for all matched extrema	1
`vardir`	Type of correlation: $+1$ for positive correlation and -1 for negative correlation; when `invornot` is set to 0, $+1$ for positive correlation and -1 for negative correlation; when `invornot` is not set to 0, this output is just a copy of `invornot`	1
`a1b` (for `alpha1best`)	if `findbest` is set to 0, `alpha1`; Otherwise, value of `alpha1` maximizing the number of found extrema	1
`a2b` (for `alpha2best`)	if `findbest` is set to 0, `alpha2`; Otherwise, value of `alpha2` maximizing the number of found extrema	1
`b1b` (for `beta1best`)	if `findbest` is set to 0, `beta1`; Otherwise, value of `beta1` maximizing the number of found extrema	1
`b2b` (for `beta2best`)	if `findbest` is set to 0, `beta2`; Otherwise, value of `beta2` maximizing the number of found extrema	1

Table C.5: Description of `PickEx.m` function outputs

C.4.2 Automatic correlation of intermediary values

Once the extrema of both variables have been matched together, either manually or with the help of function `PickEx`, intermediary values of both variables may be associated with function `InterpDis`. This code is an improvement over the analysis presented in this paper as associations are made proportionally to the distance separating two consecutive extrema along the river, when our analysis showed association made proportionally to the number of points separating two consecutive extrema disregarding the real distance between these points (see paragraph 3.3.3 on page 87). Syntax of the code is as follows:

```
[da1,exa1,da2,exa2,va1,va2,slopeall,interceptall,slopeallortho,
interceptallortho]=InterpDis(Distance1,var1,Distance2,var2,BMin,
BMax,ll1,ll2,d11,ex11,v11,d12,ex12,v12,d21,ex21,v21,d22,ex22,v22)
```

C.4.2.1 Table of inputs

Inputs of function `InterpDis` are listed in Table C.6.

Name	Description	length
`Distance1`	Distance along the channel associated with variable 1; needs to be monotonously increasing; usually same input as for `PickEx` function	L_1
`var1`	Variable 1: First variable to be correlated; usually same input as for `PickEx` function	L_1

`Distance2`	Distance along the channel associated with variable 2; needs to be monotonously increasing and unit has to be the same as that of Variable 1; usually same input as for `PickEx` function	L_2
`var2`	Variable 2: Second variable to be correlated; usually same input as for `PickEx` function	L_2
`BMin` (for `BoundaryMin`)	Distance along the channel where correlations should begin; set this parameter to any negative number to begin at the first point; usually same input as for `PickEx` function	1
`BMax` (for `BoundaryMax`)	Distance along the channel where correlations should end; to end correlations at the last point, set this parameter to a negative number or to any number smaller than `BoundaryMin`; usually same input as for `PickEx` function	1
`ll1` (for `linlog1`)	Scale of Variable 1: 0 for linear and 1 for exponential; usually same input as for `PickEx` function	1
`ll2` (for `linlog2`)	Scale of Variable 2: 0 for linear and 1 for exponential; usually same input as for `PickEx` function	1
`d11`	Distances to the minima of variable 1 whose indices are given in `ex11`; usually same as output of `PickEx` function	L_3

ex11	Indices of minima of variable 1 matched with (minima/maxima) of variable 2; usually same as output of PickEx function	L_3
v11	Values of minima of Variable 1 whose indices are given in ex11; usually same as output of PickEx function	L_3
d12	Distances to the extrema of variable 2 whose indices are given in ex12; usually same as output of PickEx function	L_3
ex12	Indices of (minima/maxima) of variable 2 matched with minima of variable 1; usually same as output of PickEx function	L_3
v12	Values of extrema of Variable 2 whose indices are given in ex12; usually same as output of PickEx function	L_3
d21	Distances to the maxima of variable 1 whose indices are given in ex21; usually same as output of PickEx function	L_4
ex21	Indices of maxima of variable 1 matched with (maxima/minima) of variable 2; usually same as output of PickEx function	L_4
v21	Values of maxima of Variable 1 whose indices are given in ex21; usually same as output of PickEx function	L_4
d22	Distances to the extrema of variable 2 whose indices are given in ex22; usually same as output of PickEx function	L_4

ex22	Indices of (maxima/minima) of variable 2 matched with maxima of variable 1; usually same as output of PickEx function	L_4
v22	Values of extrema of Variable 2 whose indices are given in ex22; usually same as output of PickEx function	L_4

Table C.6: Description of InterpDis.m function inputs

C.4.2.2 Table of outputs

Outputs of function InterpDis are listed in Table C.7.

C.4.3 Automatic plotting of regressions

After the matching process between variables, a cross-plot may be produced to show the correlations and trendlines. We also automated this part of any analysis, and the resulting code can produce a complete plot with matching points (user's choice of extrema or intermediary points), trendlines (user's choice of vertical or orthogonal least-squares), title and axis labels (units not included) and a full legend (including trendline equations corresponding to the choice of plotted points). We propose a possibility to create the plot on a grey background (vs. the usual white background) and change the font size, to help display a legible plot for presentations. Syntax is as follows:

```
CrossPlotMatches(v11,v12,v21,v22,va1,va2,ll1,ll2,nm1,nm2,vardir,
```

Name: Description	length
`da1`: Distances to the points of variable 1 whose indices are given in `exa1`	L_5
`exa1`: Indices of variable 1 points matched with points of variable 2	L_5
`da2`: Distances to variable 2 points whose indices are given in `exa2`	L_5
`exa2`: Indices of points of variable 2 corresponding to variable 1 points whose indices are given in `exa1`	L_5
`va1`: Values of Variable 1 at indices given in `exa1`	L_5
`va2`: Values of Variable 2 at indices given in `exa2`	L_5
`slall` (for `slopeall`): Slope of vertical least-square fit for all matching points	1
`icall` (for `interceptall`): Intercept (ordinate at origin) of vertical least-square fit for all matching points	1
`sloall` (for `slopeallortho`): Slope of orthogonal least-square fit for all matching points	1
`icallo` (for `interceptallortho`): Intercept (ordinate at origin) of orthogonal least-square fit for all matching points	1

Table C.7: Description of `InterpDis.m` function outputs

```
sl,ic,slo,ico,sla,ica,slao,icao,ea,vo,gp,fsize)
```

The font used in the plots will be chosen in the following order of preference and among the following families, which usually produce good-looking text:

1. Adobe Garamond Premier Pro (Semibold)

2. New Century Schoolbook

3. Century Schoolbook l

4. Times New Roman

5. Helvetica (Matlab©'s default)

C.4.3.1 Table of inputs

Inputs of function `CrossPlotMatches` are listed in Table C.8.

Name	Description	Length
`v11`	Values of minima of Variable 1 whose indices were matched to corresponding minima/maxima of Variable 2; usually same value as output of `PickEx` function	L_3
`v12`	Values of extrema of Variable 2 that correspond to values of `v11`; usually same value as output of `PickEx` function	L_3
`v21`	Values of maxima of Variable 1 whose indices were matched to corresponding maxima/minima of Variable 2; usually same value as output of `PickEx` function	L_4
`v22`	Values of extrema of Variable 2 that correspond to values of `v21`; usually same value as output of `PickEx` function	L_4

va1	Values of Variable 1 at extrema *and* intermediary points that were matched with corresponding values of Variable 2; usually same as output of InterpDis function	L_5
va2	Values of Variable 2 at extrema *and* intermediary points correspond to values of Variable 1 given in va1; usually same as output of InterpDis function	L_5
ll1 (for linlog1)	Scale of Variable 1: 0 for linear and 1 for exponential; usually same value as that used for PickEx function	1
ll2 (for linlog2)	Scale of Variable 2: 0 for linear and 1 for exponential; usually same value as that used for PickEx function	1
nm1 (for var1name)	Name of first variable, used for x-axis legend and title	(Character string)
nm2 (for var2name)	Name of second variable, used for y-axis legend and title	(Character String)
vardir	Type of correlation: $+1$ for positive correlation and -1 for negative correlation, usually obtained as result of PickEx function	1
sl (for slope)	Slope of vertical least-square fit for all matched extrema, usually obtained as result of PickEx function	1

`ic` (for `intercept`)	Intercept (ordinate at origin) of vertical least-square fit for all matched extrema, usually obtained as result of `PickEx` function	1
`slo` (for `slopeortho`)	Slope of orthogonal least-square fit for all matched extrema, usually obtained as result of `PickEx` function	1
`ico` (for `interceptortho`)	Intercept (ordinate at origin) of orthogonal least-square fit for all matched extrema, usually obtained as result of `PickEx` function	1
`sla` (for `slopeall`)	Slope of vertical least-square fit for all matching points, usually obtained as result of `InterpDis` function	1
`ica` (for `interceptall`)	Intercept (ordinate at origin) of vertical least-square fit for all matching points, usually obtained as result of `InterpDis` function	1
`slao` (for `slopeallortho`)	Slope of orthogonal least-square fit for all matching points, usually obtained as result of `InterpDis` function	1
`icao` (for `interceptallortho`)	Intercept (ordinate at origin) of orthogonal least-square fit for all matching points, usually obtained as result of `InterpDis` function	1

`ea` (for `extorall`)	-1 to plot only the extrema, $+1$ to plot intermediary points including extrema (excluding "reverse matches"), 0 to plot both extrema and intermediary points	1
`vo` (for `vertororthо`)	-1 to plot vertical least-square fit line(s), $+1$ to plot orthogonal least-square fit line(s), 0 to plot both least-square fit lines	1
`gp` (for `greyplot`)	0 displays the plot on usual, white background with black text, axes and grid lines, while 1 displays the plot on grey background with white text, axes and grid lines (better look for presentations)	1
`fsize`	Size of font (in points) used in the plot. Default value is 12, 22 or more is advised for presentations	1

Table C.8: Description of `CrossPlotMatches.m` function inputs

C.5 Notation index

ϕ: Angle between the local tangent of a curve and the mean downstream direction

ω: Maximum value of ϕ

x: In the context of *von Schelling* curves, distance along the river; otherwise abscissa in an orthonormal basis. When x is a distance along the river, $X(x)$ is the abscissa of the river at distance x

in an orthonormal basis;

y: Ordinate in an orthonormal basis. When x is a distance along the river, $Y(x)$ is the ordinate of the river at distance x in an orthonormal basis;

$x_\perp$: Distance along a *von Schelling* curve with $\omega \geq \frac{\pi}{2}$ where the first vertical tangent is encountered

t: Continuous parameterization index for parameterized curves $(x[t], y[t])$

i: Discrete parameterization index for parameterized curves $(x[i], y[i])$; used as index of datapoints on a curve

k: Index of points on left (respectively right) bank around $P_L[i]$ (resp. $P_R[i]$) within the swipe length S; consequently $-\frac{S-1}{2} \leq k \leq +\frac{S-1}{2}$ and $k = 0$ for $P_L[i]$ (resp. $P_R[i]$)

M: In the context of *von Schelling* curves, value of x corresponding to a spatial period

$R(x)$: Radius of curvature at distance x along the channel

$\gamma(x)$: Curvature at distance x along the channel, i.e. $\frac{1}{R(x)}$

$C_x[i]$: Abscissa of the center of curvature of a curve, that corresponds to point $(x[i], y[i])$

$C_y[i]$: Ordinate of the center of curvature of a curve, that corresponds to point $(x[i], y[i])$

$W(L)$: Discrete Chebyshev window of length L

$w(l)$: Discrete Ricker window of length l

$W_C(L)$: Discrete window combining $W(L)$ and $w\left(\frac{L+1}{2}\right)$ in respective proportions $1-\theta$ et θ

$P_C[i]$: Point of the centerline at index i

$P_L[i]$: Point of the left bank that is located at a minimum distance from $P_C[i]$

$P_R[i]$: Point of the right bank that is located at a minimum distance from $P_C[i]$

S: In the context of the width function algorithm, number of points around $P_L[i]$ and around $P_R[i]$ where the code looks for bank points that are the close to being on the normal to the centerline at point $P_C[i]$; In the context of Mississippi transect analysis, bottom slope;

$P_L[k]$: Point of the left bank that is located at a number of points from $P_L[i]$ smaller or equal to $\frac{S-1}{2}$

$P_R[k]$: Point of the right bank that is located at a number of points from $P_L[i]$ smaller or equal to $\frac{S-1}{2}$

$P_\star[k]$: Generic variables that designates $P_L[k]$ or $P_R[k]$

$\overrightarrow{N}[i]$: Vector normal to centerline at point $P_C[i]$

$\overrightarrow{T}[i]$: Vector tangent to centerline at point $P_C[i]$

$D_{N,L}[i,k]$: Scalar product $\overrightarrow{P_C[i]P_L[k]} \cdot \overrightarrow{N}[i]$

$D_{N,R}[i,k]$: Scalar product $\overrightarrow{P_C[i]P_R[k]} \cdot \overrightarrow{N}[i]$

$D_{N,\star}[i,k]$: Generic variable that designates $D_{N,L}[i,k]$ or $D_{N,R}[i,k]$

$D_{T,L}[i,k]$: Scalar product $\overrightarrow{P_C[i]P_L[k]} \cdot \overrightarrow{T}[i]$

$D_{T,R}[i,k]$: Scalar product $\overrightarrow{P_C[i]P_R[k]} \cdot \overrightarrow{T}[i]$

$D_{T,\star}[i,k]$: Generic variable that designates $D_{T,L}[i,k]$ or $D_{T,R}[i,k]$

$k_{max,L}$: Value of index k (depends on index i) that maximizes $D_{N,L}[i,k]$

$k_{max,R}$: Value of index k (depends on index i) that maximizes $D_{N,R}[i,k]$

$k_{max,\star}$: Generic variable that designates $k_{max,L}$ or $k_{max,R}$

$k_{sign,L}$: Value of index k (depends on index i) equal to $k_{max,L} \pm 1$ such that $D_{T,L}[i,k_{max,L}]$ and $D_{T,L}[i,k_{sign,L}]$ have opposite signs

$k_{sign,R}$: Value of index k (depends on index i) equal to $k_{max,R} \pm 1$ such that $D_{T,R}[i,k_{max,R}]$ and $D_{T,R}[i,k_{sign,R}]$ have opposite signs

$k_{sign,\star}$: Generic variable that designates $k_{sign,L}$ or $k_{sign,R}$

$N_L[i]$: Point of intersection of the left bank, linearly interpolated between existing datapoints, and the normal to the centerline at $P_C[i]$

$N_R[i]$: Point of intersection of the right bank, linearly interpolated between existing datapoints, and the normal to the centerline at $P_C[i]$

$N_\star[i]$: Generic variable that designates $N_L[i]$ or $N_R[i]$

F_L: Norm of the vector $\overrightarrow{P_C[i]N_L[i]}$

F_R: Norm of the vector $\overrightarrow{P_C[i]N_R[i]}$

$F_\star$: Generic variable that designates F_L or F_R

$W_C[i]$: Sum of the distances between $P_L[i]$ and $P_C[i]$, and between $P_R[i]$ and $P_C[i]$

$W_D[i]$: Direct distance between $P_L[i]$ and $P_C[i]$

A_1: Global accuracy indicator of the `Width` function; $A_1 = 0$ indicates that all bank points chosen to compute the width corresponding

to index $i, i \in [\![1;N]\!]$ of the centerline are exactly on the normal to the centerline at index $i, i \in [\![1;N]\!]$

A_2: Global accuracy indicator that checks for alignment (with corresponding centerline points) of points used to compute the width. $A_2 = 0$ indicates that at each index i of the centerline, the points on the interpolated left- and right-bank that are used to compute the width are perfectly aligned with the centerline point.

M_C: Length of the centerline in numbers of points

M_L: Length of the left bank in numbers of points

M_R: Length of the right bank in numbers of points

$\mathcal{D}_\lambda(M,\omega)$: Diagonal length of the period rectangle of a *von Schelling* curve (biunivocal function of ω)

$\lambda(M,\omega)$: Wavelength of a *von Schelling* curve, i.e. the length along the X-axis of a period of such a curve; that is $X(x=M) - X(x=0)$

$a(M,\omega)$: Amplitude of a *von Schelling* curve, i.e. the length along the Y-axis of a period of such a curve; that is $Y\left(x=\frac{M}{2}\right) - X(x=0)$

von Schelling curve: Curve defined by

$$\phi(x) = \omega \cdot \sin\left(2 \cdot \pi \cdot \frac{x}{M}\right)$$

Period rectangle: Rectangle whose side are the wavelength $\lambda(M,\omega)$ and amplitude $a(M,\omega)$ of one period of a spatially periodic curve; A given number of periods of this curve are fully enclosed in the same number of period rectangles.

$\mathcal{D}'_\lambda(M,\omega)$: Maximum length of a segment whose extremities are on

two opposite sides of the period rectangle of a *von Schelling* curve and that is perpendicular to a diagonal of this rectangle (not a bi-univocal function of ω)

ζ: Set of random numbers between -1 and $+1$ that follow and normal distribution

χ: Abscissæ of a *von Schelling* curve to which random Gaussian noise with a maximal amplitude of $\frac{\mathcal{D}'}{1{,}000}$ has been added along the normal to the curve at each point.

ξ: Ordinates of a *von Schelling* curve to which random Gaussian noise with a maximal amplitude of $\frac{\mathcal{D}'}{1{,}000}$ has been added along the normal to the curve at each point.

χ^{sample}: Subset of χ sampled at intervals ranging from 1 to 25 points.

ξ^{sample}: Subset of ξ sampled at intervals identical to those used for creating χ^{sample}.

X_{10}: 10^{th} percentile of variable X

X_{50}: 50^{th} percentile (median) of variable X

X_{90}: 90^{th} percentile of variable X

$P_1[i]$: Top-left point of quadrilateral fit to Mississippi transect at index i

$P_2[i]$: Bottom-left point of quadrilateral fit to Mississippi transect at index i

$P_3[i]$: Bottom-right point of quadrilateral fit to Mississippi transect at index i

$P_4[i]$: Top-right point of quadrilateral fit to Mississippi transect at index i

$d_1[i]$: Length of segment $P_1[i]P_2[i]$

$d_2[i]$: Length of segment $P_2[i]P_3[i]$

$d_3[i]$: Length of segment $P_3[i]P_4[i]$

$d_{top}[i]$: Length of segment $P_4[i]P_1[i]$

$u_1[i]$: Abscissa of $P_1[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$u_2[i]$: Abscissa of $P_2[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$u_3[i]$: Abscissa of $P_3[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$u_4[i]$: Abscissa of $P_4[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$v_1[i]$: Ordinate of $P_1[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$v_2[i]$: Ordinate of $P_2[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$v_3[i]$: Ordinate of $P_3[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$v_4[i]$: Ordinate of $P_4[i]$ in a vertical plane that contains the $P_1[i]P_2[i]P_3[i]P_4[i]$ quadrilateral

$u_j[i]$: Generic variable used with $j \in [\![1;4]\!]$

$v_j[i]$: Generic variable used with $j \in [\![1;4]\!]$

$u_j^n[i]$: Normalized version of $u_j[i]$

$v_j^n[i]$: Normalized version of $v_j[i]$

$u_j^s[i]$: Smoothed version of $u_j^n[i]$

$v_j^s[i]$: Smoothed version of $v_j^n[i]$

B: Channel width

S_2: Channel bottom slope

H: Channel height of total relief

L: Channel bottom length

S_2^s: Channel bottom slope computed from smoothed coordinates

H^s: Channel height of total relief computed from smoothed coordinates

L^s: Channel bottom length computed from smoothed coordinates

$\mathcal{G}$: Generic variable that designates B, L^s, H^s or $\left|S_2^s\right|$

$\overline{\mathcal{G}}$: Variable $\mathcal{G}$ to which a low-pass filter has been applied

α: Coefficient used to multiply $\overline{\mathcal{G}}$ when looking for extrema of $\mathcal{G}$; Adjusting α may change the number of extrema found as well as which extrema are detected

β: Coefficient used to multiply $\overline{|R|}$ when looking for extrema of $|R|$; Adjusting β may change the number of extrema found as well as which extrema are detected

$R_\perp$: Sum of the distances between points of a scatter plot and the least-square fit line

I: $R_\perp$ divided by the total number of points used to compute the least-square fit

LLD: Abrevation for *Lake Livingston Dam*

UTM-X: Universal Transverse Mercator coordinate zone X

$\mathbf{J}_n(x)$: Bessel **J** function of the first kind, of order n evaluated at x;

$$\mathbf{J}_n(x) = \frac{1}{\pi}\int_0^{\pi} \cos\left[n\cdot u - x\cdot\sin(u)\right] \mathrm{d}u$$
$$\simeq \sum_{i=0}^{+\infty} \frac{(-1)^i}{i!\cdot\Gamma(i+n+1)}\cdot\left(\frac{x}{2}\right)^{2\cdot i+n}$$

The Bessel **J** function is precoded in Maple© and Matlab©

$\mathbf{H}_n(x)$: Struve **H** function of order n evaluated at x; For $\Re(n) > -\frac{1}{2}$

$$\begin{aligned}\mathbf{H}_n(x) &= \frac{2\cdot\left(\frac{x}{2}\right)^n}{\sqrt{\pi}\cdot\Gamma\left(n+\frac{1}{2}\right)}\cdot\int_0^{\frac{\pi}{2}}\sin\left[(x\cdot\cos(u)\right]\sin^{2\cdot n}[u]\,\mathrm{d}u \\ &\simeq \sum_{i=0}^{+\infty}\frac{(-1)^i}{\Gamma\left(i+\frac{3}{2}\right)\cdot\Gamma\left(i+\frac{3}{2}+n\right)}\cdot\left(\frac{x}{2}\right)^{2\cdot i+n+1}\end{aligned}$$

The Struve **H** function is precoded in the Maple© software, and a Matlab© implementation may be found online

Bibliography

Ralph A. Bagnold. Some aspects of the shape of river meanders. Technical report, U. S. Geological Survey Professional Paper, 1960. pp. 135–144.

Vineet K. Birman, Eckart H. Meiburg, and Benjamin C. Kneller. The shape of submarine levees: exponential or power law? Journal of Fluid Mechanics, 619: 367–376, 2009.

Julian D. Clark and Kevin T. Pickering. Architectural elements and growth patterns of submarine channels: Application to hydrocarbon exploration. AAPG Bulletin-American Association of Petroleum Geologists, 80(2):194–221, 1996a.

Julian D. Clark and Kevin T. Pickering. Submarine Channels: Processes and Architecture. Vallis Press, London, 1996b.

R. K. T. Corney, J. Peakall, D. R. Parsons, L. Elliott, K. J. Amos, J. L. Best, G. M. Keevil, and D. B. Ingham. The orientation of helical flow in curved channels. Sedimentology, 53(2):249–257, 2006.

W.E. Dietrich and J.T. Perron. The search for a topographic signature of life. Nature, 439(7075):411–418, January 26 2006.

William E. Dietrich. Settling velocity of natural particles. Water Resources Research, 18(6):1615–1626, 1982.

Noritaka Endo, Tsuguo Sunamura, and Hiroshi Takimoto. Barchan ripples under unidirectional water flows in the laboratory: formation and planar morphology. Earth Surface Processes and Landforms, 30(13):1675–1682, 2005.

Ryan C. Ewing. Personal communication, 2008.

Ryan C. Ewing and Gary A. Kocurek. Aeolian dune interactions and dune-field pattern formation: White sands dune field, new mexico. Sedimentology, 57(5): 1199–1219, 2010.

R. I. Ferguson, D. R. Parsons, S. N. Lane, and R. J. Hardy. Flow in meander bends with recirculation at the inner bank. Water Resources Research, 39(11):13, 2003.

Marcelo H. García. Hydraulic jumps in sediment-driven bottom currents. Journal of Hydraulic Engineering-Asce, 119(10):1094–1117, 1993.

Marcelo H. García. Depositional turbidity currents laden with poorly sorted sediment. Journal of Hydraulic Engineering-Asce, 120(11):1240–1263, 1994.

Stanley D. Gedzelman. A universal shape for meanders. Pure and Applied Geophysics, 112(2):265–273, 1974.

F. J. Harris. Use of windows for harmonic-analysis with Discrete Fourier-Transform. Proceedings of the IEEE, 66(1):51–83, 1978.

P.F. Hudson and R.H. Kesel. Channel migration and meander-bend curvature in the lower Mississippi River prior to major human modification. Geology, 28(6): 531–534, June 2000.

Douglas J. Jerolmack and David Mohrig. A unified model for subaqueous bed form dynamics. Water Resources Research, 41(12):W12421, 2005.

Atsunari Katsuki, Hiraku Nishimori, Noritaka Endo, and Keisuke Taniguchi. Collision dynamics of two barchan dunes simulated using a simple model. Journal of the Physical Society of Japan, 74(2):538–541, 2005.

G. Kocurek, M. Townsley, E. Yeh, K. Havholm, and M. L. Sweet. Dune and dune-field development on padre-island, texas, with implications for interdune deposition and water-table-controlled accumulation. Journal of Sedimentary Petrology, 62(4):622–635, 1992.

G.M. Kondolf and M.L. Swanson. Channel adjustments to reservoir construction and gravel extraction along Stony-Creek, California. Environmental Geology, 21 (4):256–269, August 1993.

Peter Frederick Lagasse, W.J. Spitz, L.W. Zevenbergen, D.W. Zachmann, and Owen Ayres & Associates Inc. Handbook for predicting stream meander migration. Report 533, National Cooperative Highway Research Program (NCHRP), 2004.

Michael P. Lamb, Brandon McElroy, Bryant Kopriva, John Shaw, and David Mohrig. Linking river-flood dynamics to hyperpycnal-plume deposits: Experiments, theory, and geological implications. Geological Society of America Bulletin, 122(9-10):1389–1400, 2010.

Walter Basil Langbein and Luna Bergere Leopold. River meanders - theory of minimum variance. Technical report, U.S. Geological Survey, 1966.

Luna B. Leopold and M. Gordon Wolman. River meanders. Geological Society of America Bulletin, 71(6):769–793, 1960.

J. A. Nittrouer. Sediment transport dynamics in the lower Mississippi River: non-uniform flow and its effects on river-channel morphology. PhD thesis, The University of Texas, Austin, TX, 2010.

Jeffrey A. Nittrouer, David Mohrig, Mead A. Allison, and Aymeric-Pierre B. Peyret. The lowermost Mississippi River: a mixed bedrock-alluvial channel. Sedimentology, In press:no–no, May 2011. ISSN 1365-3091. doi: 10.1111/j.1365-3091.2011.01245.x. URL http://dx.doi.org/10.1111/j.1365-3091.2011.01245.x.

William R. Normark. Observed parameters for turbidity-current flow in channels, reserve fan, lake superior. Journal of Sedimentary Research, 59(3):423–431, 1989.

G. Parker, Y. Shimizu, G. V. Wilkerson, E. C. Eke, J. D. Abad, J. W. Lauer, C. Paola, W. E. Dietrich, and V. R. Voller. A new framework for modeling the migration of meandering rivers. Earth Surface Processes and Landforms, 36(1):70–86, 2011. ISSN 1096-9837. doi: 10.1002/esp.2113. URL http://dx.doi.org/10.1002/esp.2113.

Gary Parker. Morphodynamics web page, April 2006. URL http://vtchl.uiuc.edu/people/parkerg/.

Carlos Pirmez and Jasim Imran. Reconstruction of turbidity currents in Amazon channel. Marine And Petroleum Geology, 20(6-8):823–849, 2003.

Lincoln F. Pratson, Jasim Imran, Eric W. H. Hutton, Gary Parker, and James P. M. Syvitski. Bang1d:: a one-dimensional, lagrangian model of subaqueous turbid surges. Computers & Geosciences, 27(6):701–716, 2001.

Hunter Rouse. Modern conceptions of the mechanics of fluid turbulence. In Transactions of the American Society of Civil Engineers, volume 102, pages 463–541, 1937.

Kenneth Ian Skene. Architecture of submarine channel levees. PhD thesis, Dalhousie University, Halifax, Nova Scotia, Canada, 1998.

Kenneth Ian Skene, David J. W. Piper, and Paul S. Hill. Quantitative analysis of variations in depositional sequence thickness from submarine channel levees. Sedimentology, 49(6):1411–1430, 2002.

Kyle M. Straub and David Mohrig. Quantifying the morphology and growth of levees in aggrading submarine channels. Journal of Geophysical Research-Earth Surface, 113(F3):F03012, 2008.

Kyle M. Straub, David Mohrig, Brandon McElroy, James Buttles, and Carlos Pirmez. Interactions between turbidity currents and topography in aggrading sinuous submarine channels: A laboratory study. Geol Soc Am Bull, 120(3-4): 368–385, 2008.

Kyle M. Straub, David Mohrig, James Buttles, Brandon McElroy, and Carlos Pirmez. Quantifying the influence of channel sinuosity on the depositional mechanics of channelized turbidity currents: A laboratory study. Marine and Petroleum Geology, 28(3):744–760, 2011a.

Kyle M. Straub, David Mohrig, James Buttles, Brandon McElroy, and Carlos Pirmez. Quantifying the influence of channel sinuosity on the depositional mechanics of channelized turbidity currents: A laboratory study. Marine and Petroleum Geology, 28:744–760, 2011b.

Hermann von Schelling. Most frequent random walks. Technical report, General Electric Co., Adv. Technol. Lab., 1964.

R. J. Wasson and R. Hyde. A test of granulometric control of desert dune geometry. Earth Surface Processes and Landforms, 8(4):301–312, 1983.

Garnett P. Williams and M.G. Wolman. Downstream effects of dams on alluvial rivers. U.S. Geological Survey Professional Paper, 1286:83, 1984.

Index

Vita

Aymeric-Pierre B. Peyret received an engineering degree from the École Centrale de Lille, France, and a Master of Science in Petroleum Engineering at the University of Texas at Austin. He is currently a Graduate Research Assistant and a Philosophy Doctor candidate at the Jackson School of Geosciences. His professional interests include sonic-log interpretation, numerical simulation, signal processing, sediment flow modeling, geomorphology, morphodynamics, geophysics and formation evaluation.

Permanent address: 502 Lee Shore Lane
Houston, Texas 77079

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.